ISSN 2415-7740 (Print)
ISSN 2415-7074 (Online)

Belarusian State University of Informatics and Radioelectronics

Open Semantic Technologies for Intelligent Systems

Research Papers Collection

Founded in 2017

Issue 6

Minsk
2022

UDC 004.822+004.89-027.31

The collection includes peer-reviewed articles approved by the Editorial board. Designed for university professors, researchers, students, graduate students, undergraduates, as well as for specialists of enterprises in the field of intelligent systems design.

Editorial address:
220005, Minsk, Platonova str., 39, rm. 606 b

Phone: +375 (17) 293-80-92

E-mail: ostisconf@gmail.com

Web-site: ***http://proc.ostis.net***

The collection is included in the List of scientific publications of the Republic of Belarus for publishing the results of dissertation research in the technical field of science (computer science, computer technology and management)

The collection is included in the Russian Science Citation Index

ISSN 2415-7740 (Print)
ISSN 2415-7074 (Online)

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Открытые семантические технологии проектирования интеллектуальных систем

Сборник научных трудов

Основан в 2017 году

Выпуск 6

Минск
2022

УДК 004.822+004.89-027.31

Сборник включает прошедшие рецензирование и утвержденные Редакционной коллегией статьи.

Предназначен для преподавателей высших учебных заведений, научных сотрудников, студентов, аспирантов, магистрантов, а также для специалистов предприятий в сфере проектирования интеллектуальных систем.

Адрес редакции:

220005, г. Минск, ул. Платонова 39, каб. 606 б

Телефон: +375(17)293-80-92

Электронный адрес: ostisconf@gmail.com

Сайт: ***http://proc.ostis.net***

Сборник включён в Перечень научных изданий Республики Беларусь для опубликования результатов диссертационных исследований по технической отрасли науки (информатика, вычислительная техника и управление)

Сборник включён в Российский индекс научного цитирования

Министерство связи
и информатизации
Республики Беларусь

HIGH TECH PARK HTP BELARUS

BAAI

Оипи

БЕЛОРУССКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

БрГТУ БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПОЛОЦКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
имени Евфросинии Полоцкой

MCMXL

Савушкин

Qulix
SYSTEMS

ВирусБлокАда

Semantic
intelligent semantic systems

HQ
SOFTWARE

nextsoft

BIXIOM

banuba

Science Solutions
Разработка инновационных решений

FusionTech.

POLYRAN

TABLE OF CONTENTS

СОДЕРЖАНИЕ

FOREWORD

Research papers collection “Open Semantic Technology for Intelligent Systems” is devoted to the flexible and compatible technologies development that ensure the rapid and high-quality design of intelligent systems for various purposes.

The collection reflects research in the field of artificial intelligence in the following areas:

- Hybrid intelligent computer systems;
- Intelligent human-machine systems;
- Computer vision;
- Fuzzy computing;
- Intelligent agents;
- Intelligent automation;
- Knowledge management;
- Knowledge engineering;
- Ontological design;
- Semantic networks;
- Machine learning;
- Neural networks;
- Natural-language interface

The main focus of this issue is on the development of models, methods and tools that ensure the semantic compatibility of intelligent computer systems and their ability to coordinate their activities in the collective solution of complex problems.

In total, the collection contains 34 articles. The editors are thankful for all authors who sent their articles. Scientific experts selected for publication the best of the submitted works, many of them were revised in accordance with the comments of reviewers.

We are grateful our scientific experts for their great job in reviewing the articles in close cooperation with the authors. Their work allowed to raise the level of scientific results presentation, and also created a platform for further scientific discussions.

We hope that, as before, the collection will perform its main function — to promote active cooperation between business, science and education in the field of artificial intelligence.

Editor-in-chief
Golenkov Vladimir

ПРЕДИСЛОВИЕ

Сборник научных трудов «Открытые семантические технологии проектирования интеллектуальных систем» посвящен вопросам разработки гибких и совместимых технологий, обеспечивающих быстрое и качественное построение интеллектуальных систем различного назначения.

В сборнике отражены исследования в сфере искусственного интеллекта по следующим направлениям:

- Гибридные интеллектуальные компьютерные системы;
- Интеллектуальные человеко-машинные системы;
- Компьютерное зрение;
- Нечеткие вычисления;
- Интеллектуальные агенты;
- Интеллектуальная автоматизация;
- Управление знаниями;
- Инженерия знаний;
- Онтологическое проектирование;
- Семантические сети;
- Машинное обучение;
- Искусственные нейронные сети;
- Естественно-языковой интерфейс.

Основной акцент в этом выпуске сборника сделан на разработку моделей, методов и средств, обеспечивающих семантическую совместимость интеллектуальных компьютерных систем и их способность координировать свою деятельность при коллективном решении сложных задач.

В общей сложности сборник содержит 34 статьи. Редакция сборника благодарит всех авторов, представивших свои статьи. Для публикации научными экспертами были отобраны лучшие из представленных работ, многие из них были переработаны в соответствии с замечаниями рецензентов.

Мы также благодарим экспертов за большой труд по рецензированию статей в тесном взаимодействии с авторами, который позволил повысить уровень изложения научных результатов, а также создал платформу для дальнейших научных дискуссий.

Надеемся, что, как и прежде, сборник будет выполнять свою основную функцию — способствовать активному сотрудничеству между бизнесом, наукой и образованием в области искусственного интеллекта.

Главный редактор
Голенков Владимир Васильевич

Factors that determine the level of intelligence of cybernetic systems

Alexandr Zagorskiy
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: alexandr.zagorskiy.research@gmail.com

***Abstract*—In the article, a hierarchical system of properties of cybernetic systems that determine their quality and allow formulating the requirements, that a cybernetic system with strong intelligence must satisfy, is considered. The quality level of cybernetic systems is determined by a sufficiently large set of parameters of cybernetic systems. Each of the parameters determines the quality level of the cybernetic system in the corresponding aspect, indicating the level of development of specific abilities and capabilities of the cybernetic system. The obtained results will allow assessing the quality level of cybernetic systems, as well as determining the direction for development of a cybernetic system to increase the level of intelligence.**

***Keywords*—cybernetic system, cybernetic system intelligence, cybernetic system quality level, multi-agent system, computer system**

I. Introduction

At present, the usage of intelligent systems in various fields is becoming more and more relevant. The modern technology of Artificial Intelligence is represented by a variety of technologies focused on developing and maintaining various types of components of intelligent computer systems. Along with all their advantages, they have a number of serious drawbacks associated with the laboriousness of their development and maintenance. In particular, these disadvantages include the following ones:

- lack of a metric to assess the current level of intelligence of a particular system;
- lack of a standardized method for comparing the level of intelligence of different systems;
- lack of understanding of how to increase the level of intelligence of a cybernetic system.

To further increase the efficiency and level of practical significance of intelligent computer systems in increasing the level of automation of human activity [1], [2], it is necessary to clarify possible directions for improving the quality and level of intelligence in modern intelligent computer systems.

This work is dedicated to clarifying the criteria and relevant parameters that determine the quality of cybernetic systems, their level of intelligence. Quality criteria are touched upon not only for artificial cybernetic systems (computer systems of various types) but also for natural cybernetic systems, as well as various kinds of symbioses of natural and artificial cybernetic systems.

II. Problem definition

Intelligence is not defined in terms of matching human reaction times, error rates, or exact responses, but instead, the purpose is to build computer systems that exhibit the full range of the cognitive capabilities we find in humans [3]. From a formal point of view, intelligence is a set of classes of cybernetic systems, each of which includes cybernetic systems that are equivalent in terms of the level and nature for the manifestation of intelligent properties (including abilities). Thus, the nature of intelligent properties and the level of their development may be different for different cybernetic systems. Accordingly, cybernetic systems can be compared with each other.

The main property of a cybernetic system is the level of its intelligence. Intelligence is an integral characteristic that determines the level of efficiency during interaction of a cybernetic system with the environment of its existence. The process of evolution of cybernetic systems should be considered as a process of increasing the level of their quality in a number of properties and, first of all, as a process of increasing the level of intelligence of these cybernetic systems. At the same time, we can talk about the evolution of each specific cybernetic system in the course of its “life activity”, as well as the evolution of a whole class of cybernetic systems, when new instances of this class are more intelligent than their predecessors. In this aspect, the evolution of computer systems (artificial cybernetic systems) can be considered.

Various system metrics have been proposed for measuring the quality of computer-based systems. As computer-based systems grow in complexity with many subsystems or components, measuring their quality in multiple dimensions is a challenging problem [4]. Quality metrics involve a set of measures that can describe the attributes of a system in terms that are independent of the structure which leads to these attributes; these measures should be quantified numerically and should have a significant level of accuracy and reliability [5].

Researchers of artificial intelligence have traditionally defined intelligence as an inherent property of a machine

[6]. At the moment, there are no standardized metrics that allow assessing the level of intelligence of a cybernetic system, therefore, there is no way to compare different systems. The problem of identifying criteria for the intelligence of systems is considered in a number of works [7], [8], [9], [10], [11]. Nevertheless, for the practical implementation of intelligent computer systems, it is necessary to detail and refine these properties, trying to reduce them to more constructive, transparent, and understandable properties for implementation. It is important to clarify what other properties of cybernetic systems determine the level and nature of their intelligence. It is essential to clarify the point by which the level of intelligence of a cybernetic system can be increased.

III. Proposed approach

As part of this work, it is proposed to use an *OSTIS Technology* [12] as a basis, the principles of which make it possible to formally clarify and agree on the interpretation of such concepts as *cybernetic system*, *quality of a cybernetic system*, *multi-agent system*, and others within the corresponding set of ontologies. Based on the results obtained, it is necessary to clarify properties that affect the level of intelligence.

The OSTIS Technology is based on a universal way of semantic representation of information in the memory of intelligent computer systems, called an *SC-code*. SC-code texts are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, depending on orientation, can be *sc-arcs* or *sc-edges*). The *Alphabet of the SC-code* consists of five main elements, on the basis of which SC-code constructions of any complexity are built, including more specific types of sc-elements (for example, new concepts). The memory that stores SC-code constructions is called semantic memory, or *sc-memory*.

Within the technology, several universal variants of visualization of *SC-code* constructions are proposed, such as *SCg-code* (graphic variant), *SCn-code* (nonlinear hypertext variant), *SCs-code* (linear string variant).

The basis of the knowledge base within the OSTIS Technology is a hierarchical system of subject domains and ontologies. Based on this, in order to solve the above problems, it is proposed to implement a complex *Subject domain of cybernetic systems* and the corresponding *Ontology of the quality of cybernetic systems*.

Within this article, fragments of structured texts in the SCn code [13] will often be used, which are simultaneously fragments of the source texts of the knowledge base, understandable both to a human and to a machine. This allows making the text more structured and formalized, while maintaining its readability. The symbol ":=" in such texts indicates alternative (synonymous) names of the described entity, revealing in more detail certain of its features.

The development of the specified family of sc-models of subject domains and ontologies will allow:

- explicitly linking the class of the system and the parameters corresponding to this system;
- assessing the intelligence level for a system of a particular class;
- comparing systems of different classes in terms of the level of intelligence, i.e. providing an ability to solve more complex problems;
- making automation tools for assessing the intelligence level for the system.

Further, fragments of sc-models of the specified subject domains and ontologies are considered in more detail.

IV. Typology of cybernetic systems

The term "cybernetic system" has a clear quantitative definition. It is a system that dynamically matches acquired information to selected actions relative to a computational issue that defines the essential purpose of the system or machine [14]. A cybernetic system is a system that is able to control its actions, adapting to changes in the state of the external environment, the environment of its "habitat". The purpose of such adaptation can be both self-preservation (preserving own integrity and "comfort" of existence by keeping own "vital" parameters within certain limits of "comfort") and the formation of certain reactions, influences on the external environment in response to certain stimuli, situations, or events in the external environment.

The cybernetic system is able to evolve in the direction of:

- studying its external environment at least to predict the consequences of its own influences on the external environment, as well as to predict changes in the external environment that do not depend on its own influences;
- studying oneself and, in particular, its interaction with the external environment [15];
- creating technologies, methods, means, that ensure a change in their external environment, the conditions of their existence in their own interests.

The cybernetic system is an adaptive, purposeful system, an active subject of the independent life. The functioning of such a system is based on the processing of information about the environment in which this system exists. It is able to monitor, analyze, and actively influence its state and the state of the environment.

The classification of cybernetic systems on the basis of naturalness or artificiality is given bellow.

A feature of computer systems is that they can play the "role" not only of the products of the corresponding actions for the implementation of these systems, but they themselves are subjects capable of performing (automating) a wide range of actions. At the same

cybernetic system
⇒ _subdividing*_:
Attribute of naturalness or artificiality of cybernetic systems
= {• _natural cybernetic system_
⊃ _human_
• _artificial cybernetic system_
• _symbiosis of natural and artificial cybernetic systems_
⊃ _community of computer systems and humans_
}

time, the intellectualization of these systems significantly expands this spectrum.

The structural classification of cybernetic systems is given bellow.

cybernetic system
⇒ _subdividing*_:
Structural classification of cybernetic systems
= {• _simple cybernetic system_
• _individual cybernetic system_
• _multi-agent system_
}

The development level for a simple cybernetic system is below the level of individual cybernetic systems. It is a specialized means of information processing, a specialized problem solver that most often implements (interprets) one problem-solving method and, accordingly, solves only problems of a given class of problems. A simple cybernetic system can be a component built into an individual cybernetic system, and it can also be an agent of a multi-agent system, which is a collective of simple cybernetic systems.

An individual cybernetic system is a cybernetic system with a level of development based on the transition from a specialized problem solver to an individual solver that provides the interpretation of an arbitrary (non-fixed) set of problems-solving methods (programs), provided that these methods are introduced (loaded, recorded) in memory of a cybernetic system. Such a system is capable of being independent, able to "survive" on its own. The features of individual cybernetic systems are:

- the presence of memory designed to store at least interpreted methods, programs and provide correction of stored methods, their deletion from memory, and the introduction of new methods into memory;
- the ability to "re-program" a cybernetic system easily to solve other problems, which is ensured by the presence of a universal problem-solving model and, accordingly, a universal interpreter of any models represented in the corresponding language;
- the presence of even simple means of communication, information exchange with other cybernetic systems (for example, with humans);
- the ability to enter into various collectives of cybernetic systems.

A multi-agent system is a collective of interacting autonomous cybernetic systems that have a common operation environment [16]. The classification of multi-agent systems is given bellow.

multi-agent system
⇒ _subdividing*_:
{• _single-level multi-agent system_
• _hierarchical multi-agent system_
}

A single-level multi-agent system implements either one model of parallel (distributed) problem solving of the corresponding class or a combination of a fixed number of different and parallel implemented problem-solving models. A hierarchical multi-agent system consists of agents, which can be individual cybernetic systems, collectives of individual cybernetic systems, as well as collectives consisting of individual cybernetic systems and collectives of individual cybernetic systems.

V. Structure of a cybernetic system

The generalized decomposition of the cybernetic system is represented in Figure 1:

cybernetic system
⇒ _generalized decomposition*_:
{• _information stored in the memory of a cybernetic system_
• _abstract memory of a cybernetic system_
• _problem solver of a cybernetic system_
• _physical shell of a cybernetic system_
}

The information stored in the memory of a cybernetic system is an information model of the environment in which this cybernetic system operates, exists, and functions.

The abstract memory of a cybernetic system is the internal abstract information environment of a cybernetic system, which is a dynamic information construction. Each state of such an information structure is nothing but the information stored in the memory of the cybernetic system at the appropriate moment in time.

The problem solver of a cybernetic system is a combination of all the skills and abilities acquired by a

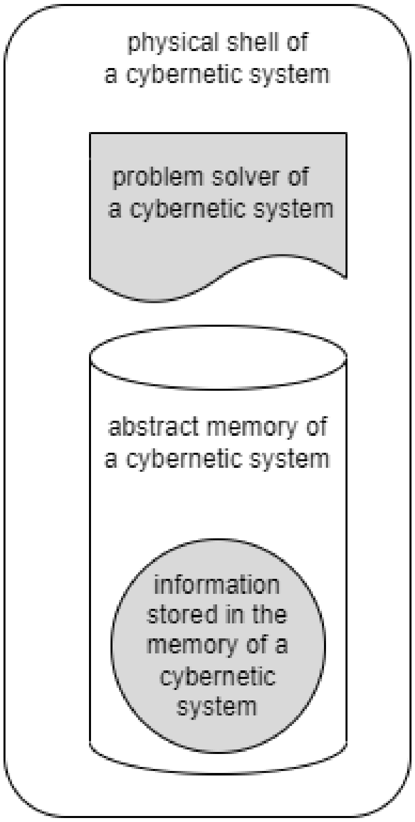


Figure 1. The structure of the cybernetic system

cybernetic system by the moment in question. A problem solver is a subject built into a cybernetic system that is capable of performing purposeful, "conscious" actions in the external environment of this cybernetic system, as well as in its internal environment (in abstract memory). The classification of actions of a cybernetic system is represented below:

action of a cybernetic system
⇒ *subdividing**:
{● *external action of a cybernetic system*
● *action of a cybernetic system performed in its own physical shell*
● *action of a cybernetic system performed in its own abstract memory*
}

Each complex action performed by a cybernetic system outside its own abstract memory includes sub-actions performed in the specified abstract memory. This means that all external actions of a cybernetic system are controlled by its internal actions (actions in abstract memory).

It should be noted that the conventionally allocated component of the cybernetic system problem solver is the interface of the cybernetic system. It provides a solution to interface problems aimed at the direct implementation of the interaction of a cybernetic system with its external environment. Such an interface should be distinguished from the physical provision of the interface of a cybernetic system.

The physical shell of a cybernetic system is a part of a cybernetic system that is an "intermediary" between its internal environment (memory in which information of a cybernetic system is stored and processed) and its external environment. The generalized decomposition of the physical shell of a cybernetic system is given bellow.

physical shell of a cybernetic system
⇒ *generalized decomposition**:
{● *cybernetic system memory*
● *cybernetic system processor*
● *physical provision of the cybernetic system interface*
● *cybernetic system physical shell*
}

In this context, the memory of a cybernetic system is understood as a physical shell, the implementation of the abstract memory of a cybernetic system, within which the cybernetic system forms and uses, processes an information model of its external environment.

Not every cybernetic system possesses memory. In cybernetic systems without memory, information processing is reduced to the exchange of signals between the components of these systems. The occurrence of memory in cybernetic systems as a medium for "centralized" storage and processing of information is the most important stage in their evolution. The fact of the occurrence of memory in a cybernetic system significantly increases the level of adaptability of a cybernetic system to various changes in its environment. The principles of organizing the memory of a cybernetic system can be different (associative, addressable, structurally fixed/structurally adjustable, non-linear/linear). Its quality largely depends on the organization of memory.

The cybernetic system processor is a hardware-implemented interpreter of the methods and programs stored in the memory of a cybernetic system, corresponding to the problem-solving model basic for this cybernetic system.

An example of the physical shell of artificial cybernetic systems is a computer. A computer for intelligent computer systems must be an effective hardware interpreter of any problem-solving models, both intelligent problems and fairly simple ones, since an intelligent system should be able to solve any problems.

VI. Complex of properties that determines the overall quality level of cybernetic systems

The quality of a cybernetic system is an integral, comprehensive assessment of the development level for a cybernetic system. This is a property, a characteristic of cybernetic systems, a sign of their classification, which allows placing these systems along the "steps" of some conditional "evolutionary ladder". Each such "step" includes cybernetic systems that have the same level of development, each of which corresponds to its own set of values of additional properties for cybernetic systems that refine (detail, specialize) the corresponding development level of cybernetic systems. Such an evolutionary approach to the consideration of cybernetic systems makes it possible, firstly, to detail the directions for evolution of cybernetic systems and, secondly, to clarify the place of this evolution, where and due to which the transition from non-intelligent cybernetic systems to intelligent ones is carried out.

The evolutionary approach to considering the diversity of cybernetic systems is based on the position that ideal cybernetic systems do not exist, but there is a constant striving for the ideal, for greater perfection. It is important to clarify what specifically in each cybernetic system should be changed in order to bring this system to a more perfect form.

So, for example, the development of computer system design technologies should be aimed at the transition to such new architectural and functional principles underlying computer systems that provide a significant reduction in the complexity of their development and reduce development time, as well as provide a significant increase in the level of intelligence and, in particular, the level of learnability of the developed computer systems, for example, by moving from supporting learning with a teacher to implementing effective self-learning (to automating the organization of self-study).

To refine the concept of the cybernetic system quality, it is necessary to set a metric of the cybernetic system quality and build a hierarchical system of properties, parameters, features, that determine the quality of cybernetic systems.

A. Complex of properties that determine the quality of the physical shell of a cybernetic system

The quality of the physical shell of the cybernetic system is the integral quality of the physical, hardware basis of the cybernetic system. The selected set of properties that determine the quality of the physical shell of a cybernetic system is given below:

It is essential that the memory provides a high level of flexibility to the specified information model. It is also important that this information model should be a model not only of the external environment of the cybernetic system but also a model of this information model itself – a description of its current situation, prehistory, regularities.

cybernetic system quality

⇒ *prerequisite property**:
- *quality of the physical shell of a cybernetic system*
- *quality of information stored in the memory of a cybernetic system*
- *quality of the problem solver of a cybernetic system*
- *hybridity of a cybernetic system*

  ⇒ *private property**:

  {
  - *variety of knowledge types stored in the memory of a cybernetic system*
  - *variety of problem-solving models*
  - *variety of types of sensors and effectors*

  }
- *adaptability of the cybernetic system to its improvement*
- *cybernetic system performance*
- *reliability of the cybernetic system*
- *cybernetic system interoperability*

quality of the physical shell of a cybernetic system

⇒ *prerequisite property**:
- *memory quality of a cybernetic system*
- *cybernetic system processor quality*
- *quality of cybernetic system sensors*
- *quality of cybernetic system effectors*
- *adaption of the physical shell of a cybernetic system to its improvement*
- *ease of transportation of the cybernetic system*
- *reliability of the physical shell of a cybernetic system*

memory quality of a cybernetic system

⇒ *prerequisite property**:
- *ability of the memory of a cybernetic system to provide storage of high-quality information*
- *ability of a cybernetic system memory to provide a high quality problem solver*
- *memory size*

The fact of the occurrence of memory in a cybernetic system is the most important stage in its evolution. Further development of the cybernetic system memory, which ensures the storage of more and more high-quality information stored in memory and more and more high-quality organization of the processing of this information, i.e. the transition to supporting more and more high-quality information processing models, is the most important factor in the evolution of cybernetic systems.

The ability of the memory of a cybernetic system to ensure the functioning of a high-quality problem solver is based on the quality of access to information stored in the cybernetic system memory, the logical and semantic flexibility of the memory of a cybernetic system, the ability of the memory of a cybernetic system to provide interpretation of a wide variety of problem-solving models.

The quality of a cybernetic system processor is determined by its ability to provide a high quality problem solver.

ability of the cybernetic system processor to ensure the functioning of a high-quality problem solver

⇒ *prerequisite property*:*

- *variety of problem-solving models interpreted by the cybernetic system processor*
- *simplicity and quality of interpretation by the system processor of a wide variety of problem-solving models*
- *providing high-quality control of information processes in memory by the cybernetic system processor*
- *cybernetic system processor speed*

The maximum level of quality of the cybernetic system processor in terms of the variety of problem-solving models interpreted by the cybernetic system processor is its universality, i.e. its fundamental ability to interpret any model for solving both intelligent and non-intelligent problems. Simplicity is determined by the degree of proximity of the interpreted problem-solving models to the "physical" level of organizing the cybernetic system processor. High-quality control of information processes in memory implies a competent combination of such aspects of process management as centralization and decentralization [17], synchrony and asynchrony, sequence and parallelism.

The quality of sensors and effectors of a cybernetic system is reduced to the variety of types of sensors and effectors in a cybernetic system, i.e. to the variety of means of perception and influence on information about the current state of the external environment and its own physical shell. The adaption of the physical shell of the cybernetic system to its improvement is determined by the flexibility and stratification of the physical shell of the cybernetic system.

B. Complex of properties that determine the quality of information stored in the memory of a cybernetic system

The quality of the information model for the "habitat" of a cybernetic system, in particular, is determined by:

- the correctness of this model, the absence of errors in it;
- the adequacy of this model;
- completeness, sufficiency of the information contained in it for the effective functioning of the cybernetic system;
- structuredness and systematization.

The most important stage in the evolution of the information model for the environment of a cybernetic system is the transition from an insufficiently complete and unsystematized information model of the environment to a knowledge base.

The correctness/incorrectness of information is the level of adequacy of the stored information of the environment in which the cybernetic system exists and the information model of which this stored information is. The consistency/inconsistency of information means the level of presence in the stored information of various types of contradictions and, in particular, errors. Errors in the stored information can be syntactic and semantic, contradicting some rules that may not be explicitly represented in memory and are considered true.

The completeness/incompleteness of information is the degree to which the information stored in the memory of a cybernetic system describes the environment of existence of this system and the methods used by it for solving problems sufficiently fully so that the cybernetic system can actually solve the entire set of problems corresponding to it. The more complete the information stored in the memory of a cybernetic system is, the more complete the information support of the activity of this system, the more effective (of higher quality) this activity itself is. The completeness is determined by the structuring of information and the variety of knowledge types stored in the memory of a cybernetic system.

The unambiguity/ambiguity of information is determined by the variety of forms of information duplication and the frequency of information duplication.

The integrity/non-integrity of information is the level of meaningful informativeness of information, the level of how semantically coherent the information is, how fully all the entities described in memory are specified (by describing the necessary set of relations of these entities with other entities being described), how rarely or often information holes occur within the stored information corresponding to the apparent insufficiency of some specifications. Examples of information holes are:

- missing method for solving common problems;

quality of information stored in the memory of a cybernetic system

⇒ *prerequisite property**:

- *semantic power of the language for representing information in the memory of a cybernetic system*
- *amount of information immersed into the memory of a cybernetic system*
- *degree of convergence and integration of various types of knowledge stored in the memory of a cybernetic system*
- *stratification of information stored in the memory of a cybernetic system*
- *simplicity and locality of performing semantically integral operations on information, stored in the memory of a cybernetic system*
- *correctness/incorrectness of information stored in the memory of a cybernetic system*
- *unambiguity/ambiguity of information stored in the memory of a cybernetic system*
- *integrity/non-integrity of information stored in the memory of a cybernetic system*
- *compliance/incompliance of information stored in the memory of a cybernetic system*
- *reliability/unreliability of information stored in the memory of a cybernetic system*
- *accuracy/inaccuracy of information stored in the memory of a cybernetic system*
- *clarity/fuzziness of information stored in the memory of a cybernetic system*
- *certainty/uncertainty of information stored in the memory of a cybernetic system*

- missing definition of the concept being used;
- insufficiently detailed specification of frequently considered entities.

The compliance/incompliance of information means the variety of forms and the total amount of information garbage that is part of the information stored in the memory of a cybernetic system. Information garbage is understood as an information fragment that is part of information, the removal of which will not significantly complicate the operation of a cybernetic system. Examples of information garbage are:

- information that is not frequently needed but can be easily inferred, when necessary;
- information that has expired.

The semantic power of the language for representing information in the memory of a cybernetic system is determined by the hybrid nature of the information stored in the memory of a cybernetic system. A language whose information constructions can represent any configuration of any relations between any entities is a universal language. The universality of the internal language of a cybernetic system is the most important factor in its intelligence.

The hybridity of information stored in the memory of a cybernetic system is determined by the variety of knowledge types and the degree of convergence and integration of various knowledge types.

Stratification of information is the ability of a cybernetic system to allocate such sections of information stored in the memory of this system that would limit the areas of action for the agents of the problem solver of the cybernetic system, which are sufficient to solve the given problems. Stratification is determined by the structuredness and reflexivity of information stored in the memory of a cybernetic system. The reflexivity of information stored in the memory of a cybernetic system, i.e. the presence of meta-linguistic means, is a factor that provides not only the structuring of stored information but the possibility of describing the syntax and semantics of the most diverse languages used by a cybernetic system.

The knowledge base is an example of information stored in the memory of a cybernetic system and that has a high quality level in all respects and, in particular, a high level of:

- semantic power of the language for representing information stored in the memory of cybernetic systems;
- hybridity of information stored in the memory of a cybernetic system;
- variety of knowledge types stored in the memory of a cybernetic system;
- formalization of information stored in the memory of a cybernetic system;
- structuredness of information stored in the memory of a cybernetic system;

The transition of information stored in the memory of a cybernetic system to a quality level corresponding to knowledge bases is the most important stage in the evolution of cybernetic systems.

C. Complex of properties that determine the quality of a problem solver of a cybernetic system

The quality of the problem solver of a cybernetic system is an integral qualitative assessment of the set of problems that a cybernetic system is capable of performing at a given moment. The main property and purpose of the problem solver of a cybernetic system is the ability to solve problems based on various types of skills accumulated and acquired by the cybernetic system using the cybernetic system processor, which is a universal interpreter of all kinds of accumulated skills. The quality of this ability is determined by a number of additional factors.

The total volume of problems solved by a cybernetic system is determined by the power of the language for representing problems solved by a cybernetic system. The power of the problem representation language is primarily determined by the variety of types of problems represented (the variety of types of described actions). Each problem is a specification of the corresponding (described) action. Therefore, consideration of the variety of types of problems solved by a cybernetic system is fully consistent with the variety of activities carried out by this system. It is important to note that there are activities of a cybernetic system that determine the quality and, in particular, the level of intelligence of a cybernetic system.

The ability of a cybernetic system to analyze the problems to be solved involves assessing the problem for:

- achievement difficulty;
- expediency of achievement (need, importance, priority);
- compliance of the purpose with the existing norms (rules) of the relevant activity.

A problem-solving method is a type of knowledge stored in the memory of a cybernetic system and containing information that is sufficient either to reduce each problem from the corresponding class of problems to a complete system of subproblems, the solution of which guarantees the solution of the initial problem, or to finally solve this problem from the specified class of problems. As problem-solving methods, not only algorithms can act but also functional programs, production systems, logical calculations, genetic algorithms, artificial neural networks of various types. Problems, for which there are no methods corresponding to them, are solved using problem-solving meta-methods (strategies) aimed at:

- generating the necessary initial data (the necessary context) for solving each problem;
- generating a plan for solving the problem, which describes the reduction of the initial problem to subproblems (down to those subproblems whose solution methods are known to the system);
- narrowing the problem solution area (to the context of the problem sufficient for its solution).

The quality of the solution of each problem is determined by:

- the time of its solution (the faster the problem is solved, the higher the quality of its solution);
- completeness and correctness of the result of solving the problem;
- memory resources spent to solve the problem (the extent of a fragment of stored information used to solve the problem);

general characteristics of the problem solver of a cybernetic system
⇒ *prerequisite property**:
{• *total volume of problems solved by a cybernetic system*
• *variety of types of problems solved by a cybernetic system*
• *ability of a cybernetic system to analyze the problems being solved*
• *ability of a cybernetic system to solve problems, methods for solving which are currently known*
• *ability of a cybernetic system to solve problems, the methods for solving which it currently does not known*
• *set of skills used by the cybernetic system*
• *degree of convergence and integration of various types of problem-solving models used by a cybernetic system*
• *quality of organizing interaction between problem-solving processes in a cybernetic system*
• *performance of the problem solver of a cybernetic system*
• *ability of a cybernetic system to solve problems involving the usage of information that has various kinds of non-factors*
• *variety and quality of solving information retrieval problems*
• *ability of a cybernetic system to generate answers to questions of various types in case when they are completely or partially absent in the current state of the information stored in memory*
• *ability of a cybernetic system to reasoning of various kinds*
• *quality of goal-setting*
• *quality of implementation of own action plans*
• *ability of a cybernetic system to localize such an area of information stored in its memory, which is sufficient to provide a solution to a given problem*
• *ability of a cybernetic system to identify the essential in the information stored in its memory*
• *cybernetic system activity*
}

- the resources of the problem solver used to solve the problem (amount of used internal agents).

Thus, improving the quality of the process of solving each specific problem, as well as each class of problems (by improving the corresponding method, in particular, the algorithm) is an important factor in improving the quality of the problem solver as a whole.

A promising option for building a problem solver for a cybernetic system is the implementation of an agent-based information processing model, i.e. construction of a problem solver in the form of a multi-agent system whose agents process information stored in the memory of a cybernetic system and are controlled by this information (more precisely, by its current state). A special place among these agents is occupied by sensory, receptor, and effector agents, which, respectively, perceive information about the current state of the external environment and influence the external environment, in particular, by changing the state of the physical shell of the cybernetic system.

The indicated agent-oriented model for organizing the interaction of problem-solving processes in a cybernetic system is, in fact, nothing more than a model of situational control for the problem-solving processes solved by a cybernetic system both in its external environment and in its memory.

The speed of the problem solver of a cybernetic system is reduced to the speed of solving problems, the speed of the problem solver, the speed of reaction of the cybernetic system to various problem situations. In many ways, the property is determined by the speed of the processor of the cybernetic system.

Examples of problems involving the usage of information that has various kinds of non-factors are the problems of design, recognition, goal-setting, prediction, etc. Most often, these are:

- vaguely formulated problems;
- problems that are solved in conditions of incompleteness, inaccuracy, inconsistency of the source data;
- problems belonging to classes of problems for which it is almost impossible to build corresponding algorithms.

These problems are characterized by:

- the inaccuracy and unreliability of source data;
- the lack of result quality criterion;
- the impossibility or high complexity of algorithm development;
- the need to take into account the context of the problem.

The ability of a cybernetic system to generate (build, synthesize, derive) answers to a variety of questions and, in particular, to questions like "what is it", to why-questions, means the ability of a cybernetic system to explain (justify the correctness) of its actions.

Independence of goal-setting is the ability of a cybernetic system to generate, initiate, and solve problems that are not subproblems initiated by external (other) subjects, as well as the ability, based on an analysis of its capabilities, to refuse to perform a problem initiated from outside, redirecting it to another cybernetic system, or on the basis of analysis of this problem itself to substantiate its inexpediency or incorrectness. Increasing the level of independence significantly expands the capabilities of the cybernetic system, i.e. the volume of those problems that it can solve not only in "ideal" conditions, but also in real, complicated circumstances. The ability of the system to adequately prioritize its purposes and not "disperse" to achieve non-priority (insignificant) purposes is the ability to analyze the expediency of activities.

The ability of a cybernetic system to identify the essential in the information stored in its memory is the ability to identify (detect, allocate) such fragments of information stored in the memory of the cybernetic system that are essential, important for achieving the corresponding purposes. The concept of an essential fragment of information stored in the memory of a cybernetic system is relative and is determined by the corresponding problem. Nevertheless, there are important permanently solved problems, in particular, the problems of analyzing the quality of information stored in the memory of a cybernetic system. The essential fragments of the stored information, allocated in the process of solving these problems, are relative not so much in relation to the problem being solved but in relation to the current state of the stored information.

The level of activity of a cybernetic system can be different for different problems being solved, for different classes of actions performed, for different types of activities. The higher the activity of a cybernetic system is, the more it manages to do, therefore, the higher its quality and efficiency. The inverse property is the concept of passivity of a cybernetic system.

D. Complex of properties that determine the level of learnability of a cybernetic system

The learnability of a cybernetic system is the ability of a cybernetic system to improve its quality, adapting to solving new problems, the quality of the internal information of its environment model, the quality of its problem solver, and even the quality of its physical shell. The ability of a cybernetic system to improve (evolve, increase its quality), to self-improve with varying degrees of independence.

The maximum level of learnability of a cybernetic system is its ability to evolve (increase the level of its quality) as quickly as possible and in any direction, i.e. the ability to quickly and without any restrictions to acquire any new knowledge and skills.

Realization of the ability of a cybernetic system to learn, i.e. to solve the permanently initiated superproblem of self-learning, imposes additional requirements on the information stored in the memory of the cybernetic system, on the problem solver of the cybernetic system, and in the future also on the physical shell of the cybernetic system.

The most important characteristic of a cybernetic system is not only what level of intelligence the cybernetic system possesses at the moment, what set of actions (problems) it is capable of performing, but also how quickly this level can be increased.

learnability of a cybernetic system
⇒ *prerequisite property**:
- *flexibility of a cybernetic system*
- *stratified cybernetic system*
- *reflexivity of a cybernetic system*
- *limited training of a cybernetic system*
- *cognitive activity of a cybernetic system*
- *self-preservation ability of a cybernetic system*

Since learning always comes down to making certain changes to the cybernetic system being trained, without a high level of flexibility of this system, there cannot be a high level of its learning. The flexibility of possible self-changes of a cybernetic system is determined by the simplicity and variety of possible self-changes of a cybernetic system.

If the cybernetic system is stratified, it becomes possible to clearly define the scope of various changes introduced into the cybernetic system, i.e. the possibility of clearly limiting those parts of the cybernetic system beyond which there is no need to take into account the consequences of the primary changes made to the system (to carry out additional changes that are the consequences of the primary changes).

The reflexivity of a cybernetic system is the ability of a cybernetic system to self-reflection. The constructive result of the reflection of a cybernetic system is the generation in its memory of the specification of various negative or suspicious features that should be taken into account in order to improve the quality of the cybernetic system. Such features (disadvantages) include identified contradictions (mistakes), identified pairs of synonymous signs, homonymous signs, information holes.

The limitation of learning a cybernetic system defines the boundary between the knowledge and skills that the corresponding cybernetic system in principle can acquire and those knowledge and skills that the specified cybernetic system will never be able to acquire. This property determines the maximum level of potential capabilities of the corresponding cybernetic system. The maximum degree of absence of restrictions in the acquisition of new knowledge and skills is the complete absence of restrictions, i.e. full universality of the capabilities of the corresponding cybernetic systems.

The cognitive activity of a cybernetic system is curiosity, activity, and independence in acquiring new knowledge and skills. It is necessary to distinguish the ability to acquire new knowledge and skills, as well as to improve them, from the desire to do so. The desire (target setting) to learn how to solve certain problems can be formulated by a cybernetic system either independently or from outside (by some teacher).

cognitive activity of a cybernetic system
⇒ *prerequisite property**:
- *ability of a cybernetic system to synthesize cognitive purposes and procedures*
- *ability of a cybernetic system to self-organize its own learning*
- *ability of a cybernetic system to perform experimental actions*

The ability of a cybernetic system to synthesize cognitive purposes and procedures is the ability to plan its own learning and manage the learning process, the ability to ask questions or purposeful sequences of questions, the ability to generate a clear specification of its information needs. The ability of a cybernetic system to self-organize its own learning is the ability to manage its own learning, the ability of a cybernetic system itself to play the role of its teacher. The ability of a cybernetic system for experimental actions is the ability to deviate from the prepared plans of its actions in order to improve the quality of the result or maintain the purposefulness of these actions, the ability to improvise.

The higher the level of security of a cybernetic system, the higher its level of learning. The ability of a cybernetic system to self-preserve means the ability of a cybernetic system to identify and eliminate threats aimed at reducing its quality and even destroying it, which means a complete loss of the required quality.

E. Complex of properties that determine the level of intelligence of a cybernetic system

The main property, characteristic of a cybernetic system is the level of its intelligence, which is an integral characteristic that determines the efficiency level of interaction of a cybernetic system with the environment of its existence. The process of evolution of cybernetic systems should be considered as a process of increasing the level of their quality in a number of properties and as a process of increasing the level of their intelligence.

A cybernetic system can be both intelligent and non-intelligent. In turn, the intelligent system can be both low-intelligent and high-intelligent.

intelligence level of a cybernetic system
⇒ *prerequisite property**:
- *education of the cybernetic system*
- *learnability of a cybernetic system*
- *interoperability of a cybernetic system*

The level of education of a cybernetic system is the level of skills, as well as other knowledge acquired by a cybernetic system by a given moment.

education of the cybernetic system
⇒ *prerequisite property**:
- *quality of skills acquired by a cybernetic system*
- *quality of information stored in the memory of a cybernetic system*

Examples of an educated cybernetic system are:

- a knowledge-based cybernetic system;
- a knowledge-driven cybernetic system;
- a targeted cybernetic system;
- a hybrid cybernetic system;
- a potentially universal cybernetic system.

A learnable cybernetic system is a cybernetic system capable of knowing its environment, that is, building and constantly updating in its memory an information model of this environment, as well as using this model to solve various problems (to organize its activities) in this environment. Examples of a trainable cybernetic system are:

- a cybernetic system with a high level of stratification of its knowledge and skills;
- a reflexive cybernetic system;
- a self-learning cybernetic system;
- a cybernetic system with a high level of cognitive activity.

The intelligence of a cybernetic system, as well as the underlying cognitive process performed by a cybernetic system, has a social character, since it is most effectively formed and developed in the form of interaction of a cybernetic system with other cybernetic systems. A socially oriented cybernetic system has a sufficiently high level of intelligence to be a useful member of various, including human-machine communities. A certain level of socially significant qualities is a necessary condition for the intelligence of a cybernetic system. Examples of a socially oriented cybernetic system are:

- a cybernetic system capable of establishing and maintaining a high level of semantic compatibility and mutual understanding with other systems;
- a negotiable cybernetic system.

All properties inherent in cybernetic systems can have very different levels in different cybernetic systems. Moreover, in some cybernetic systems, some of these properties may not exist at all. At the same time, in cybernetic systems, which we will be conditionally called intelligent systems, all the above properties must be represented in a sufficiently developed form.

VII. Complex of properties that determine the quality of a multi-agent system

The transition from cybernetic systems to collectives of interacting cybernetic systems, i.e. to the social organization of cybernetic systems, is the most important factor in the evolution of cybernetic systems. A cybernetic system, which is a collective of interacting cybernetic systems with a certain degree of independence (self-sufficiency, freedom of choice), will be called a multi-agent system [18].

cybernetic system
⇒ *subdividing**:
{
- *individual cybernetic system*
- *cybernetic system, which is the minimum component of an individual cybernetic system*
- *cybernetic system, which is a complex of components of the corresponding individual cybernetic system*
- *community of individual cybernetic systems*
  ⇒ *subdividing**:
  {
  - *simple community of individual cybernetic systems*
  - *hierarchical community of individual cybernetic systems*

  }

}

The agents of a multi-agent system can (but do not have to) be intelligent systems. For example, agents of an intelligent problem solver with an agent-oriented architecture are not intelligent systems. An agent of a hierarchical multi-agent system may act as another multi-agent system [19].

In a multi-agent system with centralized control, there are specially allocated agents that make decisions in a certain area of activity of the multi-agent system and ensure the implementation of these decisions by controlling the activities of other agents that are part of this system.

In a multi-agent system with decentralized control [20], decisions are made collegially and “automatically” (decisions on recognizing new information proposed by someone including initiating a certain problem, decisions on correcting, clarifying previously recognized, agreed information) on the basis of a well-thought-out and constantly improved methodology, as well as on the basis of the active participation of all agents in the formation of new proposals to be recognized or agreed upon. In such a multi-agent system, all agents participate in the control of this system. An example of such a system is an orchestra capable of playing without a bandmaster.

The transition to multi-agent systems is the most important factor in improving the quality (and, in particular, the level of intelligence) of cybernetic systems, since the level of intelligence of a multi-agent system can be much higher than the level of intelligence of each agent included in it. This does not always occur, since the most important factor in the quality of multi-agent systems is not only the quality of the agents included in it but also the organization of the interaction of agents and, in particular, the transition from centralized to decentralized control. Quantity does not always become a new quality.

The quality of individual cybernetic systems is determined, among other things, by how much an individual cybernetic system contributes to improving the quality of the collectives to which it belongs. This property of individual cybernetic systems will be called the level of their interoperability [21].

A synergetic cybernetic system is a multi-agent system with a high level of collective intelligence, whose atomic agents are individual intelligent systems with a high level of interoperability [22] [23]. An example of a synergetic cybernetic system is a creative collective implementing a complex science-intensive project.

The effectiveness of the creative collective (for example, in the field of scientific and technical activities) is determined by:

- the consistency of motivation, goal-setting of the whole collective and each of its members (there should be no contradictions between the purpose of the collective and the creative self-realization of each of its members);
- the effective organization of decentralized control of the activities of collective members;
- the clear, prompt, and accessible to all documentation of the current state of the completed tasks and directions for its further development;
- the level of labor intensity of the efficiency for fixing individual results within a collectively created overall result;
- the level of structuredness and, above all, stratification of the generalized documentation (knowledge base);
- efficiency of associative access to documentation fragments;
- flexibility of a collectively created base;
- automation of analysis of the completed tasks and project control.

The intelligence level of a multi-agent system can be significantly lower than the intelligence level of the most “stupid” member of this collective, but it can also be significantly higher than the intelligence level of the most “intelligent” member of the specified collective. In order for the number of intelligent systems to turn into a significantly more intelligent quality of a collective of such systems, all intelligent systems combined into a collective must have a high level of interoperability, which imposes additional requirements on the information stored in memory, as well as on the problem solvers of intelligent systems, combined in a collective.

The interoperability of a cybernetic system is the ability of a cybernetic system to interact with other cybernetic systems in order to create a collective of cybernetic systems (multi-agent systems), the level of quality and, in particular, the level of intelligence of which is higher than the quality level of each cybernetic system that is part of this collective.

In order for the number of members of the collective of a cybernetic system to turn into a higher quality of the collective itself, the members of the collective must have additional abilities, which we will call the properties of interoperability. The main such properties are the ability to establish and maintain a sufficient level of semantic compatibility (mutual understanding) with other cybernetic systems and negotiability (the ability to coordinate own actions with others) [24].

Purposeful exchange of information between cybernetic systems significantly accelerates the process of their learning (the process of accumulating knowledge and skills). Consequently, the ability to effectively use the specified channel for the accumulation of knowledge and skills significantly increases the level of learning of cybernetic systems. Increasing the level of interoperability of a cybernetic system is, on the one hand, an additional increase in the level of intelligence of this cybernetic system itself, as well as a factor in increasing the level of intelligence of those collectives, those multi-agent systems that include this cybernetic system.

cybernetic system interoperability
⇒ *prerequisite property**:
- *negotiability of a cybernetic system*
- *social responsibility of a cybernetic system*
- *social activity of a cybernetic system*

Properties-preconditions for the level of negotiability of a cybernetic system are represented below:

Understanding information coming from outside includes:

negotiability of a cybernetic system

⇒ *prerequisite property**:

- *the ability of a cybernetic system to understand received messages*
- *the ability of a cybernetic system to form transmitted messages understandable to recipients*
- *the ability of a cybernetic system to provide semantic compatibility with partners*
- *communication skills of a cybernetic system*
- *the ability of a cybernetic system to discuss and agree on the purposes and plans of collective activity*
- *the ability of a cybernetic system to take on the solving of urgent problems within agreed plans for collective activity*

- translation of this information into the internal language of the cybernetic system;
- local verification of input information;
- immersion (convergence, placement) of the text resulting from the specified translation into the stored information (in particular, into the knowledge base).

The immersion of the input information into the knowledge base of a cybernetic system is reduced to the identification and elimination of contradictions that arise between the immersed text and the current state of the knowledge base. The complexity of the problem of understanding the input verbal information lies not only in the complexity of the consistent immersion of the input information in the current state of the knowledge base but also in the complexity of translating this information from the external language into the internal language of the cybernetic system, i.e., in the complexity of generating the text of the internal language, semantically equivalent the input text of the external language. For natural languages, this translation is a difficult problem, since at present the problem of formalizing the syntax and semantics of natural languages has not been solved.

The semantic compatibility of two given cybernetic systems is determined by the consistency of the concepts systems used by both interacting cybernetic systems. The problem of providing permanent support for the semantic compatibility of interacting cybernetic systems is a necessary condition for ensuring a high level of mutual understanding of cybernetic systems and, as a result, their effective interaction.

The sociability of a cybernetic system is the ability of a cybernetic system to establish mutually beneficial contacts with other cybernetic systems (including collectives of intelligent systems) by honestly identifying mutually beneficial common purposes, interests.

Properties-prerequisites for the level of social responsibility of a cybernetic system are represented below:

social responsibility of a cybernetic system

⇒ *prerequisite property**:

- *the ability of a cybernetic system to fulfill its obligations in a quality and timely manner within the relevant collectives*
- *the ability of a cybernetic system to adequately assess its capabilities in the distribution of collective activity*
- *altruism/selfishness of a cybernetic system*
- *absence/presence of actions that, due to the illiteracy of the cybernetic system, reduce the quality of the collectives it is part of*
- *absence/presence of "conscious", motivated actions that reduce the quality of collectives, which include a cybernetic system*

Properties-prerequisites for the level of social activity of a cybernetic system are represented below:

social activity of a cybernetic system

⇒ *prerequisite property**:

- *the ability of a cybernetic system to generate proposed purposes and plans for collective activity*
- *activity of the cybernetic system in the examination of the results of other participants in the collective activity*
- *the ability of a cybernetic system to analyze the quality of all the collectives it belongs to, as well as all members of these collectives*
- *the ability of a cybernetic system to participate in the formation of new collectives*
- *quantity and quality of those collectives in which the cybernetic system is or was a part*

The formation of a specialized collective of cybernetic systems comes down to the fact that in the memory of each cybernetic system included in the collective, the specification of this collective is generated, which includes:

- a list of all collective members;
- abilities of each member of the collective;
- their responsibilities within the collective;

- specification of the entire set of problems (type of activity) for the solution of which the given collective of cybernetic systems is formed.

Each cybernetic system can be part of a large number of collectives, while performing, in the general case, different "duties", different "business processes" in different collectives.

VIII. Conclusion

To increase the level of automation of human activity, it is necessary to automate more complex problems. The solution of such problems is reduced to the requirement to increase the level of intelligence of individual cybernetic systems. However, the individual intelligence of cybernetic systems has its limitations. It is possible to achieve a significant increase in the level of intelligence by forming collectives of cybernetic systems, i.e. move to multi-agent systems.

Thus, one of the most important factors determining the level of intelligence of a cybernetic system is interoperability, the ability to form collectives with individual cybernetic systems. At the same time, the level of quality of individual cybernetic systems of the entire collective should be quite high. This will allow the number of intelligent systems to be transferred into a significantly more intelligent quality of the collective of such systems.

Acknowledgment

The author would like to thank the research group of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for its help in the work and valuable comments.

References

[1] E. Popkova and B. Sergi, *Artificial Intelligence: Anthropogenic Nature vs. Social Origin*, 01 2020.

[2] T. Choudhury, B. K. Dewangan, R. Tomar, B. K. Singh, T. T. Toe, and N. G. Nhu, *Autonomic Computing in Cloud Resource Management in Industry 4.0*. Springer, 2021.

[3] J. E. Laird, R. E. Wray III, R. P. Marinier III, and P. Langley, "Claims and challenges in evaluating human-level intelligent systems," in *2nd Conference on Artificiel General Intelligence (2009)*. Atlantis Press, 2009, pp. 80–85.

[4] J.-H. Cho, S. Xu, P. M. Hurley, M. Mackay, T. Benjamin, and M. Beaumont, "Stram: Measuring the trustworthiness of computer-based systems," *ACM Comput. Surv.*, vol. 51, no. 6, feb 2019.

[5] Y. S. Sherif, E. Ng, and J. Steinbacher, "Computer software development: Quality attributes, measurements, and metrics," *Naval Research Logistics (NRL)*, vol. 35, no. 3, pp. 425–436, 1988.

[6] R. Gao and L. Tsoukalas, "Performance metrics for intelligent systems: An engineering perspective," *NIST SPECIAL PUBLICATION SP*, pp. 5–10, 2002.

[7] M. Molina, "What is an intelligent system?" p. 16, 2022.

[8] Finn V.K, *Intellekt, informatsionnoe obshchestvo, gumanitarnoe znanie i obrazovanie*, Moscow, 2021.

[9] N. J. Nilsson, "Human-level artificial intelligence? be serious!" *AI magazine*, vol. 26, no. 4, pp. 68–68, 2005.

[10] C. I. Kerr, L. Mortara, R. Phaal, and D. Probert, "A conceptual model for technology intelligence," *International Journal of Technology Intelligence and Planning*, vol. 2, no. 1, pp. 73–93, 2006.

[11] S. Antsyferov, "Estimation of quality level of intellectual systems," *Iskusstvennyj intellekt [Artificial intelligence]*, 2013.

[12] Vladimir Golenkov and Natalia Guliakina and Daniil Shunkevich, *Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[13] (2022, Nov) Ostis Metasystem. [Online]. Available: https://ims.ostis.net

[14] R. L. Fry, "The engineering of cybernetic systems," in *AIP Conference Proceedings*, vol. 617, no. 1. American Institute of Physics, 2002, pp. 497–528.

[15] D. Weyns, B. SCHMERL, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. AN-DERSSON, H. Giese, and K. GOSCHKA, "Software engineering for self-adaptive systems ii, lncs vol. 7475," 2013.

[16] M. Hadzic, E. Chang, P. Wongthongtham, and T. Dillon, *Ontology-based multi-agent systems*. Springer, 2009.

[17] O. Melekhova, J. Malenfant, R. Mescheriakov, and A. Chueshev, "A decentralised solution for coordinating decisions in large-scale autonomic systems," in *MATEC Web of Conferences*, vol. 161. EDP Sciences, 2018, p. 03024.

[18] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *Ieee Access*, vol. 6, pp. 28 573–28 593, 2018.

[19] J. Ferber, O. Gutknecht, and F. Michel, "From agents to organizations: an organizational view of multi-agent systems," in *International workshop on agent-oriented software engineering*. Springer, 2003, pp. 214–230.

[20] P. G. Balaji and D. Srinivasan, "An introduction to multi-agent systems," in *Innovations in multi-agent systems and applications-1*. Springer, 2010, pp. 1–27.

[21] A. M. Ouksel and A. Sheth, "Semantic interoperability in global information systems," vol. 28, no. 1, p. 5–12, mar 1999. [Online]. Available: https://doi.org/10.1145/309844.309849

[22] P. Lopes de Lopes de Souza, W. Lopes de Lopes de Souza, and R. R. Ciferri, "Semantic interoperability in the internet of things: A systematic literature review," in *ITNG 2022 19th International Conference on Information Technology-New Generations*, S. Latifi, Ed. Cham: Springer International Publishing, 2022, pp. 333–340.

[23] Hamilton, Gunther, Drummond, and Widergren, "Interoperability - a key element for the grid and der of the future," in *2005/2006 IEEE/PES Transmission and Distribution Conference and Exhibition*, 2006, pp. 927–931.

[24] F. W. Neiva, J. M. N. David, R. Braga, and F. Campos, "Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature," *Information and Software Technology*, vol. 72, pp. 137–150, 2016.

Факторы, определяющие уровень интеллекта кибернетических систем

Загорский А.Г.

В работе рассмотрена иерархическая система свойств кибернетических систем, определяющих их качество и позволяющих сформулировать требования, которым должна удовлетворять кибернетическая система с сильным интеллектом. Уровень качества кибернетических систем определяется достаточно большим набором параметров кибернетических систем. Каждый из параметров определяет уровень качества кибернетической системы в соответствующем аспекте, указывая уровень развития конкретных способностей и возможностей кибернетической системы. Полученные результаты позволят оценить уровень качества кибернетических систем, а также определить направление развития кибернетической системы для повышения уровня интеллекта.

Received 29.10.2022

Next-generation intelligent computer systems and technology of complex support of their life cycle

Vladimir Golenkov
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
Email: golen@bsuir.by

Natalya Gulyakina
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
Email: guliakina@bsuir.by

***Abstract*—The paper considers the principles of building next-generation intelligent computer systems, as well as the principles of building a comprehensive technology for their development and life cycle support - OSTIS Technology. Semantic compatibility and interoperability are highlighted as the key properties of the next-generation intelligent systems. The paper considers an approach to providing these properties, realized within the framework of OSTIS Technology.**

***Keywords*—OSTIS, ostis-system, ontological approach, intelligent computer system, interoperability, knowledge base, problem-solving model, semantic representation of information, SC-code**

I. Introduction

The most important direction of increasing the level of *intelligence* of *individual intelligent cybernetic systems* is the transition to *individual intelligent cybernetic systems collectives* and further to *hierarchical intelligent cybernetic systems collectives*, whose members are both *individual intelligent cybernetic systems* and *individual intelligent cybernetic systems collectives*, as well as *hierarchical intelligent cybernetic systems collectives*. Similarly, it is necessary to increase the level of intelligence and *individual intelligent computer systems* (artificial *cybernetic systems*). But at the same time, we must remember that not every association of *intelligent cybernetic systems* (including *computer systems*) becomes an *intelligent collective*. For this, it is necessary to comply with additional requirements imposed on all members of *intelligent collectives*. The most important of them is the requirement of a high level of ***interoperability***, in other words the ability to interact effectively with other members of the team. The transition from modern *intelligent computer systems* to *interoperable intelligent computer systems* is a key factor in the transition to the ***next-generation intelligent computer systems***, providing a significant increase in the level of automation of human activity.

II. Next-generation intelligent computer systems

The creation of various complexes of interacting *intelligent computer systems* requires improving the quality of not only these systems themselves, but also the quality of their interaction. *Next-generation intelligent computer systems* should have a high level of ***interoperability***, in other words, a high level of ability to effectively, purposefully interact with their own kind and with users in the process of collective (distributed) and decentralized solution of complex problems [1], [2], [3], [4], [5], [6]. The level of ***interoperability*** of *intelligent computer systems* is, figuratively speaking, the level of their "socialization", usefulness within the framework of various a priori unknown communities (collectives) of *intelligent systems*. The level of ***interoperability*** of *intelligent computer systems* is the level of their communication (social) compatibility, which allows them to independently form collectives of *intelligent computer systems* and their users, as well as independently coordinate and coordinate their activities within these collectives when solving complex tasks in partially predictable conditions. Increasing the level of *interoperability* of intelligent computer systems determines the transition to the ***next-generation intelligent computer systems***, without which it is impossible to implement projects such as smart-enterprise, smart-hospital, smart-city, smart-society [7], [8].

intelligent computer system
:= [intelligent artificial cybernetic system]
⇒ *decomposition**:
{• *individual intelligent computer system*
• *intelligent collective of intelligent computer systems*
}

Let's take a closer look at the concept of intelligent collective of intelligent computer systems.

intelligent collective of intelligent computer systems

:= [intelligent multi-agent system, whose agents are intelligent computer systems]

⇒ *note**:

[Not every collective of intelligent computer systems can be intelligent, because the level of intelligence of such a collective is determined not only by the level of intelligence of its members, but also by the efficiency (quality) of their interaction.]

⇒ *decomposition**:

{
- *intelligent collective of individual intelligent computer systems*
- *hierarchical intelligent collective of intelligent computer systems*
  - := [intelligent collective of intelligent computer systems, at least one of whose members is an intelligent collective of intelligent computer systems]

}

next-generation intelligent computer systems

⇒ *requirements**:
- high level of *interoperability*
- high level of *learning*
- high level of *hybridity*
- a high level of ability to solve *intelligent problems* (in other words *problems*, which solution *methods* and/or the background information required for their solution are priori unknown)
- high level of *synergy*

interoperability^

:= [ability to effectively (focused) interaction with other independent subjects]

:= [ability to the partnership in solving *complex problems*, that require *collective activity*]

:= [ability to work in collective (in a team)]

:= [level of socialization]

:= [social skills]

high level of interoperability

⇒ *provided by**:
- high level of *understanding*
  - ⇒ *provided by**:
    - high level of ***semantic compatibility*** of a given subject with other subjects of a given collective
    - high level of *ability to understand* the messages and behavior of partners
    - high level of *ability to be understandable* for partners:
      - – ability to clearly and reasonably formulate their suggestions and information useful for solving current problems;
      - – ability to act and comment on their actions so that they and their motives are clear to partners;
    - high level of ability to increase the level of semantic compatibility with their partners
- high level of *negotiability*, in other words the ability to coordinate with partners their plans and intentions in order to ensure timely high quality of the collective result
- high level of *ability to decentralize coordination* of their actions with the actions of partners in unpredictable (abnormal) circumstances
- high level of *ability to minimize the negative consequences of conflict situations* with other subjects
  - ⇒ *provided by**:
    - high level of *ability to prevent the occurrence of conflict situations*
    - *compliance with ethical standards* and rules that prevent the occurrence of destructive consequences of conflict situations
    - high level of *ability to share responsibility* with partners for timely and high-quality achievement of a common goal

semantic compatibility^

:= [degree of coherence (coincidence) of systems of *concepts* and other *key entities*, used by specified interacting entities]

⇒ *note**:

[Provision of *semantic compatibility* requires formalization of the *semantic representation of information*]

ability to share responsibility with partners, which is a necessary condition for decentralized management of collective activity

⇒ *provided by**:
- *ability to monitor* and analyze collectively performed activity
- *ability to promptly inform partners* about adverse situations, events, trends, as well as initiate appropriate collective actions

learning^

:= [ability to quickly and efficiently acquire new *knowledge* and *skills*, as well as improve already acquired *knowledge* and *skills*]

high level of learning

⇒ *provided by**:

- high level of *flexibility of the information* stored in the memory of the intelligent system
- high level of *quality* of *stratification of information*, stored in the memory of the intelligent system by the stratification of *the knowledge base*
- high level of *reflexivity* of intelligent system
- high level of *ability to correct its mistakes* (including to eliminate contradictions in its *knowledge base*)
- high level of *cognitive activity*
- absence of *restrictions on the type of acquired knowledge and skills* (the absence of such restrictions means the potential *universality* of the intelligent system)

hybridity^

:= [degree of diversity of the *types of knowledge* and *models of problem solving* used and the level of efficiency of their sharing]

:= [individual ability to solve *complex problems* that require the use of different *types of knowledge*, as well as various combinations of different *models of problem solving*]

high level of hybridity

⇒ *provided by**:

- high degree of diversity of the *types of knowledge* and *models of problem solving* used
- high degree of *convergence* [9], [10] and deep *integration* (degree of interpenetration) of various *types of knowledge* and *models of problem solving*
- ability to indefinitely expand the level of its *hybridity*

We emphasize that the *hybridity* and *interoperability* of *intelligent computer systems* of the next-generation implies the rejection of the well-known paradigm of "black boxes", since:

- all the variety of problem solving models of *a hybrid intelligent computer system* should be interpreted on one common *universal platform*;
- availability of information about how each method used, the model of problem solving, each subject significantly improves the quality of their *coordination* when *solving complex tasks together*;
- it becomes possible to use some methods, models of problem solving and entire subjects (for instance, *intelligent computer systems*) to improve (improve the quality) of other methods, models and subjects.

It is especially necessary to note the following characteristics of *intelligent computer systems of the next-generation*:

- ***degree*** of ***convergence***, unification and standardization of *intelligent computer systems* and their components and the corresponding ***degree of integration*** (integration depth) of *intelligent computer systems* and their components;
- ***semantic compatibility*** between *intelligent computer systems* in general and *semantic compatibility* between the components of each *intelligent computer system* (in particular, compatibility between different *types of knowledge* and different *models of knowledge processing*), which are the main indicators of the degree of ***convergence*** (convergence) between *intelligent computer systems* and their components.

The feature of these characteristics of *intelligent computer systems* and their components is that they play an important role in solving all the key tasks of the modern stage of *artificial intelligence* development and are closely related to each other.

It should also be noted that the listed requirements for the *next-generation intelligent computer systems* are aimed at overcoming the curse of *Babel* [11] both within *intelligent computer systems of the next-generation* (between internal *information processes* for solving various problems) and between interacting independent *next-generation intelligent computer systems* in the process of collective solution of *complex problems*.

At the present stage of evolution of *intelligent computer systems* for a significant expansion of their application areas and a qualitative increase in the level of automation of human activity:

- It is necessary to move to the creation of semantically compatible ***intelligent computer systems of the next-generation***, focused not only on individual, but also on collective (joint) solution of *complex problems* requiring coordinated activity several independent intelligent computer systems and the use of various models and methods in unpredictable combinations, which is necessary to significantly expand the scope of application of *intelligent computer systems*, for the transition from automation of local types and areas of *human activity* to complex automation of larger (combined) types and areas of this activity;
- It is necessary to develop a ***General formal theory and standard for the next-generation intelligent computer systems***;
- It is necessary to develop a ***Technology for integrated support of the life cycle of the next-generation intelligent computer systems***, which includes support for the *engineering* of these systems (as the initial stage of their life cycle) and ensuring their *compatibility* at all stages of their life cycle;
- ***Convergence*** and ***unification*** of the *next-generation intelligent computer systems* is necessary;
- It is necessary to implement "seamless", "diffuse",

interpenetrating, ***deep integration of semantically adjacent components of intelligent computer systems***, in other words, integration in which there are no clear boundaries ("seams") of the integrated (connected) components, and which can be carried out automatically. This means moving to ***hybrid intelligent computer systems***;
- It is necessary to observe the ***Occam's razor principle*** – maximum possible structural simplification of *intelligent computer systems of the next-generation*, exclusion of eclectic solutions;
- It is necessary to focus on potentially ***universal*** (in other words, capable of quickly acquiring any knowledge and skills), ***synergetic*** *intelligent computer systems* with "strong" intelligence;

next-generation intelligent computer systems

⇒ *underlying principles**:

- *semantic representation of knowledge* in the memory of *intelligent computer systems*, assuming the absence of *homonymic signs*, which in different contexts denote different entities, as well as the absence of *synonymy*, in other words pairs of synonymous *signs*, which denote the same entity. The semantic representation of the information structure in general has a nonlinear (graph) character and is called *semantic network*
- use of a common for all intelligent computer systems *universal language of semantic representation of knowledge* in memory *intelligent computer systems*, having the simplest possible *syntax*, providing the representation of any *types of knowledge* and having unlimited possibilities of transition from *knowledge* to *meta-knowledge*. The simplicity of the syntax of *information constructions* of the specified *language* allows us to call these constructions *refined semantic networks*
- *structurally tunable (graphodynamic) memory organization* of intelligent computer systems, in which knowledge processing is reduced not so much to changing the state of stored *characters*, as to changing the configuration of the connections between these *characters*
- *semantically unlimited associative access to information* stored in the memory of *intelligent computer systems*, according to a given sample of arbitrary size and arbitrary configuration
- *decentralized situational management of information processes* in the memory of *intelligent computer systems*, realized using *agent-oriented model of knowledge base processing*, in which *initiation* of new *information processes* is not carried out by transferring control to the corresponding a priori known procedures, and as a result of the occurrence of the corresponding *situations* or *events in the memory of an intelligent computer system*, because «The main problem of computer systems is not the accumulation of knowledge, but the ability to activate the necessary knowledge in the process of solving problems» (Pospelov D.A.). Such a multi-agent process of information processing is a *activity* performed by some collective of independent *information agents* (information processing agents), the condition for initiating each of which is the appearance in the current state of *knowledge base* corresponding to this agent *situation* and/or *events*.

  The choice of multi-agent technologies is explained by the fact that currently any complex production, logistics or other system can be represented by a set of interactions of simpler systems to any level of detail, which provides a fractal-recursive principle of building multi-tiered systems built as open digital colonies and AI ecosystems. Multi-agent technologies are based on a distributed or decentralized approach to problem solving, in which dynamically updated information in a distributed network of intelligent agents is processed directly from agents along with locally available information from "neighbors". At the same time, both resource and time costs for communication in the network are significantly reduced, as well as time for processing and decision-making in the center of the system (if there is one).»

  ⇐ *quotation**:
  Barinov I.I.. DevelSFoftheConAI-2021/p. 270 [12]
- Transition to *semantic models of problem solving*, which are based on taking into account not only syntactic (structural) aspects of the processed information, but also semantic (semantic) aspects of this information – "From data science to knowledge science"
- ***ontological model of knowledge bases*** of *intelligent computer systems*, in other words the ontological structuring of all information stored in memory of *intelligent computer system*, assuming a clear *stratification of the knowledge base* in the form of a hierarchical system of *subject areas* and the corresponding *ontologies*, each of which provides a semantic *specification* of all *concepts* that are key within the corresponding *subject area*
- ***ontological localization of problem solving***

in *intelligent computer systems*, assuming localization *scope* of each stored in memory *method* and each *information agent* in accordance with *ontological model* processed by *knowledge base*. Most often, such a *scope* is one of *subject areas* or one of *subject areas* together with its corresponding *ontology*

- ***ontological model of the interface*** *intelligent computer system* which includes:
  – ontological description of *syntax* of all *languages* used by *intelligent computer system* for *communication* with external *subjects*;
  – ontological description of *denotational semantics* of each *language* used by *intelligent computer system* for *communication* with external *subjects*;
  – family of *information agents* providing *syntactic analysis*, *semantic analysis* (translation into an internal semantic language) and *understanding* (immersion in *knowledge base*) of any entered *message* belonging to any *to an external language*, the full ontological description of which is in the knowledge base of *intelligent computer system*;
  – family of *information agents* providing the *synthesis of messages* that (1) are addressed to external subjects with whom an *intelligent computer system* communicates, (2) are *semantically equivalent* to the specified *fragments of the knowledge base* of an intelligent computer system that determine the *meaning* of the transmitted *messages*, (3) belong to one of the *external languages*, the full ontological description of which is in the *knowledge base* of an intelligent computer system;
- *semantically friendly nature of the user interface*, provided by (1) a formal description in the knowledge base of the user interface management tool and (2) the introduction of the corresponding help subsystems into the *intelligent computer system*, providing a significant reduction in the language barrier between users and *intelligent computer systems*, which will significantly increase the efficiency of *operation of intelligent computer systems*
- *minimizing the negative impact of the human factor* on the efficiency of *operation intelligent computer systems* thanks to the implementation of an interoperable (partner) style of interaction not only between *intelligent computer systems* themselves, but also between *intelligent computer systems* and their users. Responsibility for the quality of joint activities should be distributed among all partners
- ***multimodality*** (hybrid character) *intelligent computer system*, which assumes:
  – variety of *types of knowledge* included in the *knowledge base* of an intelligent computer system;
  – variety of *problem solving models* used by the *problem solver* of an intelligent computer system;
  – variety of *sensor channels* that *monitor* the state of the *external environment* of an intelligent computer system;
  – variety of *effectors* that *affect* the *external environment*;
  – variety of *languages of communication* with other subjects (with users, with intelligent computer systems);
- ***internal semantic compatibility*** between the components of an *intelligent computer system* (i.e., the maximum possible introduction of common, matching *concepts* for various fragments of the stored *knowledge base*), which is a form of ***convergence*** and *deep integration* within an *intelligent computer system* for different types of *knowledge* and different *models of problem solving*, which ensures the effective implementation of the *multimodality of an intelligent computer system*
- ***external semantic compatibility*** between various *intelligent computer systems*, expressed not only in the commonality of *concepts* used, but also in the commonality of basic *knowledge* and is a necessary condition for ensuring a high level of *interoperability* of intelligent computer systems
- orientation to the use of *intelligent computer systems* as *cognitive agents* in ***hierarchical multi-agent systems***
- ***platform independence* of intelligent computer systems**, assuming:
  – clear *stratification* of each *intelligent computer system* (1) to a logical-semantic model represented by its *knowledge base*, which contains not only *declarative knowledge*, but also knowledge having *operational semantics*, and (2) to a *platform* providing *interpretation* of the specified *logical-semantic model*;
  – universality of the specified *platform* of interpretation of the *logical-semantic model of an intelligent computer system*, which makes it possible for each such *platform* to provide interpretation of any *logical-semantic model of an intelligent computer system*, if this model is presented in the

same *universal language of semantic representation of information*;
  - variety of options for implementing *platforms* for *interpreting logical and semantic models of intelligent computer systems* - both options that are programmatically implemented on *modern computers*, and options that are implemented in the form of *new-generation mainframe computers* focused on use in intelligent computer systems of a new generation (such computers we called *associative semantic computers*);
  - easily implemented possibility of transferring (reinstalling) the logical-semantic model (*knowledge base*) of any *intelligent computer system* to any other *platform for interpreting logical-semantic models*;
- initial orientation of *intelligent computer systems of the new generation* to the use of ***universal associative semantic computers*** (computers of the new generation) as a *platform for the interpretation of logical-semantic models* (knowledge bases) of *intelligent computer systems*

Currently, a large number of different types of *problem solving models*, models of representation and processing of knowledge of various types have been developed. But different combinations of these models may be in demand in different *intelligent computer systems*. When developing and implementing various *intelligent computer systems*, appropriate methods and tools should guarantee the *logical and semantic compatibility* of the components being developed and, in particular, their ability to use common *information resources*. For this, obviously, the *unification* of these models is necessary.

The variety of different types of intelligent computer systems and, accordingly, the variety of combinations of knowledge representation models and problem solving used by them is determined by:

- variety of the purpose of intelligent computer systems and the type of their environment;
- variety of different types of stored knowledge; a variety of knowledge processing models and problem solutions;
- variety of different types of interfaces (signal processing, audio, video, effector means).

The following aspects of the *compatibility* of knowledge representation and processing models in *intelligent computer systems* should be highlighted:

- syntactic;
- semantic (consistency of systems of concepts);
- functional (operational).

It should also be distinguished:

- *compatibility* between components of *intelligent computer systems*;
- *compatibility* between the upper logical-semantic level of the knowledge representation and processing models used and various levels of their interpretation up to the hardware level;
- *compatibility* between individual intelligent computer systems;
- *compatibility* between individual intelligent computer systems and their users;
- *compatibility* between teams of intelligent computer systems.

III. Semantic representation of information

semantic representation of information
:= [recording (representation) of the information structure at the semantic level]
:= [information construct whose syntactic structure is close to its meaning, i.e. close to the described configuration of connections between the described entities]
:= [semantic representation of the information structure]
⊂ *semantic network*
  ⊃ *refined semantic network*

refined semantic network
⇒ *the underlying principles**:
- Each element (syntactically atomic fragment) of a refined semantic network is a sign of one of the described entities
- Each entity described by a refined semantic network should be represented by its own sign, which is an element of this network
- Within each separate refined semantic network, there is no synonymy of different signs, and there are also no homonymous signs
- The variety of entities described by refined semantic networks is not limited by anything. Accordingly, the semantic typology of the elements of refined semantic networks is very rich
- A special type of elements of refined semantic systems are signs of connections between other elements of these networks. At the same time, connected elements (i.e. elements that are incident to the specified signs of connections) there may also be signs of other connections. Most often, the sign of the connection between the elements of a refined semantic network is a reflection of the connection between entities that are designated by these elements. But in some cases, the sign of the connection between the elements of a refined semantic network can

be a reflection, for example, of the connection between one described entity and the sign of another described entity

IV. Multi-agent models of problem solving based on the semantic representation of information

next-generation intelligent computer systems problem solver

⇒ *requirements**:

- problem solver of intelligent computer systems of a new generation should be able to solve intellectual problems, which include the following types of tasks:
  - poorly formulated problem
    := [problem whose formulation contains various non-factors (incompleteness, vagueness, inconsistency (incorrectness),..)]
  - problem for which, in addition to the formulation of the problem itself and the corresponding method of solving it, additional, but a priori unknown, information about the objects specified in the formulation (statement) of the problem is needed. At the same time, the specified additional information may or may not be present in the current state of the knowledge base of intelligent computer systems. In addition, for some tasks, the area of the knowledge base can be specified (specified), the use of which is sufficient to search for or generate (in particular, logical output) the specified additional required information. Such an area of the knowledge base will be called the area of solving the corresponding problem
  - problem for which the appropriate method of solving it is not currently known. reformulate the problem, in other words generate (logically output) a logically equivalent formulation of the original problem for which the method of its solution is currently known; To solve such a problem , you can:
    * reformulate the problem, in other words generate (logically output) a logically equivalent formulation of the original problem for which the method of its solution is currently known;
    * reduce the original problem to a family of subtasks for which the methods of their solution are currently known.
- process of solving problems in intelligent computer systems of a new generation is implemented by a team of information agents processing the knowledge base of intelligent computer systems
- management of information processes in the memory of intelligent computer systems of a new generation is carried out in a decentralized manner according to the principles of situational management

situational management

:= [situational and event management]

⇒ *explanation**:

[managing the sequence of actions, in which the condition ("trigger") for initiating these actions is:

□ occurrence of some situations (conditions, states);

□ and/or the occurrence of some *events*

]

situation

:= [structure describing some temporarily existing configuration of relationships between some entities]

:= [description of the temporarily existing state of some fragment (some part) of some dynamic system]

event

⊃ *emergence of a temporary entity*
 := [appearance, birth, the beginning of the existence of some temporary entity]

⊃ *disappearance of a temporary entity*
 := [termination, termination of the existence of some temporary entity]

⊃ *transition from one situation to another*
 ⇒ *note**:
 [It takes into account not only the fact of the emergence of a new situation, but also its background - i.e. the situation that immediately precedes it. So, for example, reacting to an abnormal value of a parameter, it is important for us to know:

 □ what is the dynamics of the change of this parameter (it increases or decreases and at what rate);

 □ what measures were taken earlier to eliminate this anomaly.

 ]

V. Ontological models of interfaces of intelligent computer systems based on the semantic representation of information

interface of the next-generation intelligent computer system

⇒ *underlying principles**:

- interface of an *intelligent computer system of a new generation* is considered as a solver of a particular type of problems - *interface problems*, the main of which are:
  - problems of understanding verbal information acquired by an intelligent computer system (syntactic analysis, semantic analysis and immersion in the knowledge base of an intelligent computer system)
  - problems of understanding nonverbal information perceived by sensory subsystems of an intelligent computer system (image analysis, audio signal analysis, immersion of analysis results into the knowledge base of an intelligent computer system)
  - problems of synthesizing messages addressed to external entities (cybernetic systems)
- fact that the interface of an *intelligent computer system of a new generation* is a solver of a particular type of *problems of an intelligent computer system of a new generation*, the properties underlying the *problem solvers of intelligent computer systems of a new generation* are inherited by the interfaces of *intelligent computer systems of a new generation*. It follows from this that the basis of *intelligent computer systems of the new generation* is:
  - semantic representation of accumulated (acquired knowledge);
  - interpretation of semantic analysis of acquired verbal information as a process of translating this information into the internal language of the semantic representation of knowledge, followed by immersion (input, integration) of the result of this translation into the current state of the knowledge base of an *intelligent computer system of a new generation*;
  - interpretation of the synthesis of messages addressed by external entities as a process of reverse translation of some fragment of the knowledge base from the internal language of the semantic representation of information into an external language used to communicate with a given subject;
  - agent-oriented organization of interface problem solving, implemented by the respective teams of internal agents of the interface of *intelligent computer systems of a new generation* interacting through the knowledge base of an *intelligent computer system of a new generation* that is publicly available to them
- interface of an *intelligent computer system of a new generation* is interpreted as a specialized integrated *intelligent computer system of a new generation*, which is part of the above-mentioned intelligent computer system, the knowledge base of which includes:
  - ontology of the syntactic internal language of the semantic representation of information;
  - ontology of denotational semantics of the internal language of semantic representation of information;
  - syntax ontology of all external languages used to communicate with external entities;
  - ontologies of denotational semantics of all external languages used for communication with external subjects (each such ontology from a formal point of view is a description of the correspondence between the texts of external languages and semantically equivalent texts of the internal language of the semantic representation of information.

    At the same time, we emphasize that all of these ontologies, which are part of the knowledge base, interfaces of intelligent computer systems of a new generation, as well as all other information included in this knowledge base, are presented in the internal language of the semantic representation of information, which, accordingly, is used in this case as a meta language.

Conversations about a friendly and, in particular, adaptive *user interface* have been going on for a long time, but this most often concerns the form ("syntactic" side) of the *user interface*, and not the semantic content of interaction with users. Currently, *user interfaces* of computer systems (including *intelligent computer systems*) for a wide contingent of users are not semantically (meaningfully) friendly (semantically comfortable). The organization of user interaction with computer systems (including *intelligent computer systems*) is a "bottleneck" that has a significant impact on the efficiency of *automation of human activity*. The modern organization of user interaction with a computer system is based on the paradigm of a competent user who knows what he wants from the tool he uses and is fully responsible for the quality of interaction with this tool. This paradigm underlies the activities of a logger in interaction with an axe, a rider in interaction with a horse, a car driver, a pilot in interaction with a corresponding vehicle, an operator of a nuclear power plant, a railway dispatcher, and so on.

At the present stage of the development of *Artificial intelligence*, in order to increase the efficiency of interaction, it is necessary to move from the paradigm of competent management of the tool used to the paradigm of equal cooperation, partnership interaction of an *intelligent computer system* with its user. An *intelligent computer system* should turn "face" to the user.

Semantic friendliness of the user interface should consist in adaptability to the peculiarities and qualifications of the user, the exclusion of any problems for the user in the process of dialogue with an *intelligent computer system*, in permanent care of improving the user's communication skills.

VI. Advantages of the proposed approach to the creation of intelligent computer systems of a new generation what measures were taken earlier to eliminate this anomaly

The ***semantic representation of information*** in the memory of *intelligent computer systems* ensures the elimination of duplication of information stored in the memory of an *intelligent computer system*, i.e. the elimination of a variety of forms of representation of the same information, the prohibition of the appearance in one memory of *semantically equivalent information structures*, including synonymous *signs*. This significantly reduces complexity and improves quality:

- development of various *models of knowledge processing* (because there is no need to take into account the variety of forms of representation of the same knowledge);
- *semantic analysis* and *understanding* of information received (transmitted) from various external entities (from users, from developers, from other *intelligent computer systems*);
- *convergence* and *integration* of different types of knowledge within each *intelligent computer system*;
- ensuring *semantic compatibility* and *mutual understanding* between various *intelligent computer systems*, as well as between *intelligent computer systems* and their users.

We consider the concept of *semantic network* not as a beautiful metaphor of complexly structured *sign constructions*, but as a formal refinement of the concept of *semantic representation of information*, as the principle of representation of information underlying a new generation of *computer languages* and *computer systems* themselves - *graph languages* and *graph computers*. A *semantic network* is a nonlinear (graph) *sign construction* with the following properties:

- all elements (in other words syntactically elementary fragments) of this *graph structure* (nodes and bundles) are *signs* of the entities being described and, in particular, *signs of connections* between these entities;
- all *signs* included in this *graph structure* do not have *synonyms* within this structure;
- "internal" pattern (structure) of the *signs* included in the *semantic network* does not need to be taken into account in its *semantic analysis* (understanding);
- meaning of a *semantic network* is determined by the *denotational semantics* of all the *signs* included in it and the configuration of the *incidence relationships* of these signs;
- of the two *incident signs* included in the *semantic network*, at least one is a communication sign.

A *refined semantic network* is a *semantic network* having a simple *syntactic structure* in which, in particular,

- finite *alphabet* of *semantic network* elements is used, in other words a finite number of syntactically distinguished types (syntactic labels) attributed to these elements;
- external identifiers (in particular, names) attributed to *semantic network* elements are used only for information input/output.

An *agent-oriented model of information processing* combined with *decentralized situational control of the information processing process*, as well as with the *semantic representation of information* in the memory of an *intelligent computer system* significantly reduces the complexity and improves the quality of integration of various *models of problem solving* in the processing of a common *knowledge base*. This refers to the simultaneous use of different *models of problem solving* when processing the same *knowledge*, in particular, when solving the same *problem*.

A high level of *semantic flexibility of information* stored in the memory of an *intelligent computer system of a new generation* is ensured by the fact that each deletion or addition of a syntactically elementary fragment of stored information, as well as the deletion or addition of each *incident relationship* between such elements has a clear semantic interpretation.

A high level of *stratification of information* stored in the memory of the *next-generation intelligent computer system* is provided by the ontologically oriented structuring of the knowledge *base of the next-generation intelligent computer system.*

A high level of *individual learning* of the next-generation intelligent computer systems (in other words their ability to rapidly expand their *knowledge* and *skills*) is provided:

- *semantic flexibility of the information* stored in their *memory*;
- *the stratification* of this information;
- *reflexivity* of intelligent computer systems of the new generation.

A high level of *collective learning* of the next-generation intelligent computer systems is ensured by a high level of their *socialization* (i.e. the ability to effectively participate in the activities of various collectives consisting of the *next-generation intelligent computer systems* and people) and, above all, a high level of their *mutual understanding*.

The high level of *interoperability* of intelligent computer systems of the new generation fundamentally changes the nature of the interaction of *computer systems* with people whose activities they automate - from the management of these automation tools to *equal partner meaningful relationships*.

Each *next-generation intelligent computer system* is capable of:

- independently or by invitation to join a team consisting of *intelligent computer systems of a new generation* and/or people. Such teams are created on a temporary or permanent basis for the collective solution of complex *tasks*;
- participate in the distribution (including coordination of the distribution) of *tasks* – both "one-time" tasks and long-term tasks (responsibilities);
- monitor the state of the entire process of collective activity and coordinate their activities with the activities of other members of the team in case of possible unpredictable changes in the conditions (state) of the relevant environment.

The high level of *intelligence of the next-generation intelligent computer systems* and, accordingly, the high level of their independence and purposefulness allows them to be full members of a wide variety of communities within which *next-generation intelligent computer systems* receive the right to initiate independently (based on a detailed analysis of the current state of affairs and, including, the current state of the community action plan) a wide the range of actions (problems) performed by other members of the community, and thus participate in the coordination and coordination of the activities of community members. The ability of the *next-generation intelligent computer system* to coordinate its activities with other similar systems, as well as to adjust the activities of the entire *collective of the next-generation intelligent computer systems*, adapting to various types of changes in the environment (conditions) in which this activity is carried out, allows you to significantly automate the activities of a *system integrator* both at the stage of creating a *collective of the next-generation intelligent computer systems*, and and at the stage of its updating (reengineering).

The advantages of *intelligent computer systems of the new generation* are provided by:

- advantages of the language of internal *semantic encoding of information* stored in the memory of these systems;
- advantages of the organization of graphodynamic associative semantic memory of *intelligent computer systems of a new generation*;
- advantages of *semantic representation of knowledge bases* of intelligent computer systems of a new generation and *means of ontological structuring of knowledge bases* of these systems;
- advantages of *agent-oriented problem solving models* used in *intelligent computer systems of a new generation* in combination with decentralized control of the information processing process.

VII. Technology of integrated life cycle support for intelligent computer systems of the next generation

life cycle of the next-generation intelligent computer system

⇒ *includes**:

- designing the next-generation intelligent computer system

  ⇒ *includes**:

  - designing the knowledge base of the next-generation intelligent computer system
  - designing the next-generation intelligent computer system problem solver
  - designing the interface of the next-generation intelligent computer system
- realization of the next-generation intelligent computer system
- initial training of the next-generation intelligent computer system
- quality monitoring of the next-generation intelligent computer system
- restoring the required level of the next-generation intelligent computer system
- reengineering of the next-generation intelligent computer system
- ensuring the security of the next-generation intelligent computer system
- operation of the next-generation intelligent computer system by end users

The construction of a *technology* for *integrated support of the life cycle of the next-generation intelligent computer systems* involves:

- Clear description of the current version of the *next-generation intelligent computer systems standard*, which ensures semantic compatibility of the systems being developed;
- Creation of powerful libraries of semantically compatible and reusable components of developed *intelligent computer systems*;

- Clarification of the requirements for the integrated technology being created and caused by the features of the *next-generation intelligent computer systems* developed and operated using this technology.

Creation of an infrastructure that provides intensive permanent development of *Technology* for *integrated support of the life cycle of the next-generation intelligent computer systems* involves:

- Ensuring a low threshold of entry into the *technology of designing intelligent computer systems* for both technology users (i.e. developers of applied or specialized intelligent computer systems) and developers of the technology itself;
- Ensuring high rates of *technology* development by taking into account the experience of developing various applications by actively involving application authors to participate in the development (improvement) of *technology*.

At the heart of the creation of the ***technology*** we offer ***for integrated support of the life cycle of intelligent computer systems of the next generation***, are the following provisions:

- implementation of the proposed *technology* for the development and maintenance of *intelligent computer systems of the next generation* in the form of an ***intelligent computer metasystem*** that fully complies with the *standards* of the proposed *intelligent computer systems of the next generation* developed by the proposed *technology*. The structure of such an *intelligent computer metasystem* implementing the proposed technology includes:
  - formal ontological description of the current version of the *standard for intelligent computer systems of the next generation*;
  - formal ontological description of the current version of *methods and tools for designing, implementing, maintaining, reengineering and operating intelligent computer systems of the next generation*.

  Due to this, the technology of designing and reengineering intelligent computer systems of a new generation and the technology of designing and reengineering the technology itself (i.e. intelligent computer metasystem) are the same thing;
- ***unification*** and ***standardization*** **of intelligent computer systems of the next generation**, as well as *methods* of their *design, implementation, maintenance, reengineering and operation*;
- permanent evolution of the ***standard of intelligent computer systems of the next generation***, as well as *methods* of their *design, implementation, maintenance, reengineering and operation*;
- ***ontological design*** **of intelligent computer systems of the next generation**, assuming:
  - clear coordination and operational formalized fixation (in the form of *formal ontologies*) of the approved *current state* of the hierarchical system of all *concepts* underlying the permanently evolving *standard of intelligent computer systems of the next generation*, as well as at the heart of each developed *intelligent computer system*;
  - fairly complete and prompt documentation of the current status of each project;
  - using the *"top-down"* design *methodology*.
- ***component design*** of intelligent computer systems of a new generation, i.e. design focused on the assembly of *intelligent computer systems* from ready-made components based on constantly expanding libraries of *reusable components*;
- ***complex nature*** of the proposed *technology* that performs:
  - support for *designing* not only *components* of *intelligent computer systems of the next generation* (various *fragments of knowledge bases, knowledge bases* in general, various *methods of problem solving*, various *internal information agents, problem solvers* in general, formal ontological descriptions of various *external languages*, *interfaces* in general), but also *intelligent computer systems* in general as independent *design objects* taking into account the specifics of those classes to which the designed *intelligent computer systems* belong;
  - support not only for the *integrated design* of *intelligent computer systems* of the *next generation*, but also support for their implementation (assembly, reproduction), maintenance, reengineering during operation and operation itself.

To create a *technology* for integrated design and comprehensive support for the subsequent stages of the life cycle of *intelligent computer systems of the next generation*, it is necessary:

- Unify the formalization of various models of representation of various types of used information stored in the memory of *intelligent computer systems* and various models of solving intelligent problems to ensure *semantic compatibility* and simple automated integrability of various types of *knowledge* and *models of solving problems* in *intelligent computer systems*. To do this, it is necessary to develop a basic *universal* abstract model of knowledge representation and processing, which provides the implementation of various models of problem solving.
- Unify the structuring of *knowledge bases* of intelligent computer systems in the form of a hierarchical system of ontologies of different levels.
- Unify the system of *concepts* used, specified by the corresponding *ontologies* to ensure *semantic compatibility* and *interoperability* of various *intelligent computer systems*.

- Unify the architecture of *intelligent computer systems*, providing *semantic compatibility*:
  - between *intelligent computer systems* and their users;
  - between *individual intelligent computer systems*;
  - between *collective intelligent computer systems*,

  as well as ensuring the *interoperability* of communities consisting of:
  - *individual intelligent computer systems*;
  - *collective intelligent computer systems*;
  - users of intelligent computer systems
- To develop a *basic model of interpretation* of various formal models of problem solving in intelligent computer systems with a focus on the maximum possible simplification of such interpretation in *next generation computers* that are specifically designed for the implementation of individual *intelligent computer systems*.
- To develop the *next generation of computers*, the principles of functioning of which are as close as possible to the basic abstract model, which ensures the integration of all kinds of knowledge representation models and problem solving models. At the same time, the basic information processing machine underlying these computers should differ significantly from the von Neumann machine and should be close to the basic model of problem solving in intelligent computer systems in order to significantly reduce the complexity of interpreting the entire variety of problem solving models in intelligent computer systems.

The implementation of all these stages of the development of *Artificial Intelligence technologies* represents a transition to a fundamentally new technological order, which provides a significant increase in the efficiency of practical use of the results of work in the field of *Artificial intelligence* and a significant increase in the level of automation of *human activity*.

We have called the proposed *technology of integrated support of the life cycle of the next-generation intelligent computer systems* the ***OSTIS Technology*** (Open Semantic Technology for Intelligent Systems). Accordingly, *intelligent computer systems of the next generation* developed using this technology are called ***ostis-systems***. The *OSTIS technology* itself is implemented by us in the form of a special *ostis-system*, which we call the ***OSTIS Metasystem*** and the *knowledge base* of which contains:

- Formal theory of *ostis-systems*;
- Standard of *ostis-systems*
  - Standard for *ostis-systems* knowledge base
    - Standard of the internal universal language of semantic representation of knowledge in the memory of *ostis-systems*
    - Standard for the internal representation of top-level ontologies in the memory of *ostis-systems*
    - Standard for the presentation of the source texts of knowledge bases of *ostis-systems*
  - Standard for *ostis-systems* problem solvers
    - Standard of the *ostis-systems* basic programming language
    - Standard of high-level programming languages for *ostis-systems*
    - Standard for the representation of artificial neural networks in the memory of *ostis-systems*
    - Standard of internal information agents in *ostis-systems*
  - Standard for *ostis-systems* interfaces
    - The standard of external languages of *ostis-systems* close to the internal universal language of semantic representation of knowledge
- Standard for ostis-systems and the OSTIS Technology (***OSTIS standard***) [13];
- The core of the Library of reusable components *ostis-systems* (***OSTIS Libraries***);
- Methods and *tools to support the life cycle* of *ostis-systems* and their components.

ostis-system

⇒ *subdividing**:

{• *ostis-subject*

:= [independent *ostis-system*]

⇒ *subdividing**:

{• *individual ostis-system*

• *collective ostis-system*

:= [multi-agent system, which is a collective of individual and collective *ostis-systems*, whose activities are coordinated by the corresponding corporate *ostis-system*]

⇒ *note**:

[*The ostis-systems* collective may include individual *ostis-systems* may include individual *ostis-systems* of any kind – including corporate *ostis-systems* representing the interests of other *ostis-systems* teams]

}

- *built-in ostis-system*
  := [*ostis-system*, which is part of some individual *ostis-system*]

}

individual ostis-system
:= [minimal independent *ostis-system*]
⇒ *subdividing**:
{• *personal ostis-assistant*
  := [*ostis-system*, which provides comprehensive adaptive service to a specific user on *all* issues related to his interaction with any other *ostis-systems*, as well as representing the interests of this user in the entire global network of *ostis-systems*]
- *corporate ostis-system*
  := [*ostis-system* that coordinates the joint activities of *ostis-systems* within the framework of the corresponding *ostis-system* collective, monitors and reengineers the corresponding set of *ostis-systems* and represents the interests of this collective within other *ostis-system* collectives]
- *individual ostis system that is neither a personal ostis assistant nor a corporate ostis system*

}

VIII. Conclusion

Let us briefly list the main provisions of this work:

- The main practically significant direction of the development of modern *intelligent computer systems* is the transition to *interoperable intelligent computer systems* capable of effective interaction with each other and with users, which
  - provides automation of solving complex problems that require the creation of temporary or permanent collectives
  - turns *intelligent computer systems* into independent active *subjects* capable of initiating various complex problems and, in fact, initiating for this purpose workable collectives consisting of people and *interoperable intelligent computer systems* of the required qualifications
- Collectives consisting of independent *interoperable intelligent computer systems* and people have good prospects of becoming synergetic systems.
- The *interoperability of intelligent computer systems* is ensured
  - a high level of mutual understanding and, accordingly, semantic compatibility
  - a high level of contractual capacity, in other words. the ability to pre-coordinate their actions with the actions of other subjects
  - a high level of ability to quickly coordinate their actions with the actions of other subjects in the course of their realization
- Among the principles underlying the construction of *interoperable intelligent computer systems* are:
  - semantic representation of knowledge in the memory of *intelligent computer systems* in the form of refined semantic networks
  - using the universal language of the internal semantic representation of knowledge
  - graphodynamic organization of knowledge processing
  - agent-based problem solving models
  - structuring and stratification of knowledge bases in the form of a hierarchical system of formal ontologies
  - semantically friendly user interface
- To develop a large number of interoperable semantically compatible *intelligent computer systems* that ensure the transition to a fundamentally new level of automation of *human activity*, it is necessary to create technologies that ensure the mass production of such *intelligent computer systems*, participation in which is available to a wide contingent of developers (including developers of intermediate qualifications and novice developers). The main provisions of this technology are
  - standardization of interoperable *intelligent computer systems*
  - widespread use of *component design* based on a powerful library of semantically compatible reusable (typical) components of *interoperable intelligent computer systems*
- Effective operation of *interoperable intelligent computer systems* requires the creation of not only the *technology of designing* such systems, but also a family of technologies to support all other stages of their life cycle. This is especially true of the technology of permanent support of*semantic compatibility* of *all* interacting *interoperable intelligent computer systems* during their operation.

Acknowledgment

The authors would like to thank the scientific team of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for their assistance and valuable comments.

References

[1] K. Yaghoobirafi and A. Farahani, “An approach for semantic interoperability in autonomic distributed intelligent systems,” *Journal of Software: Evolution and Process*, vol. 34, no. 10, p. e2436, 2022.

[2] Ouksel, A. M. and Sheth, A., “Semantic interoperability in global information systems,” *SIGMOD Rec.*, vol. 28, no. 1, p. 5–12, mar 1999.

[3] Lanzenberger, Monika and Sampson, Jennifer and Kargl, Horst and Wimmer, Manuel and Conroy, Colm and O’Sullivan, Declan and Lewis, David and Brennan, Rob and Ramos-Gargantilla, José Ángel and Gómez-Pérez, Asunción and Fürst, Frédéric and Trichet, Francky and Euzenat, Jérôme and Polleres, Axel and Scharffe, François and Kotis, Konstantinos, “Making ontologies talk: Knowledge interoperability in the semantic web,” *IEEE Intelligent Systems*, vol. 23, no. 6, pp. 72–85, 2008.

[4] Francila Weidt Neiva and José Maria N. David and Regina Braga and Fernanda Campos, “Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature,” *Information and Software Technology*, vol. 72, pp. 137–150, 2016.

[5] J. Pohl, “Interoperability and the need for intelligent software: A historical perspective,” 09 2004.

[6] Jeff Waters and Brenda J. Powers and Marion G. Ceruti, “Global interoperability using semantics, standards, science and technology (gis3t),” *Computer Standards & Interfaces*, vol. 31, no. 6, pp. 1158–1166, 2009.

[7] Lopes de Lopes de Souza, Pedro and Lopes de Lopes de Souza, Wanderley and Ciferri, Ricardo Rodrigues, “Semantic interoperability in the internet of things: A systematic literature review,” in *ITNG 2022 19th International Conference on Information Technology-New Generations*, Latifi, Shahram, Ed. Cham: Springer International Publishing, 2022, pp. 333–340.

[8] Hamilton and Gunther and Drummond and Widergren, “Interoperability - a key element for the grid and der of the future,” in *2005/2006 IEEE/PES Transmission and Distribution Conference and Exhibition*, 2006, pp. 927–931.

[9] A. E. Yankovskaya, A. A. Shelupanov, A. N. Kornetov, N. N. Ilinskaya, and V. B. Obukhovskaya, “Gibridnaya intellektual’naya sistema ekspress-diagnostiki organizatsionnogo stressa, depressii, deviantnogo povedeniya i trevogi narushitelei na osnove konvergentsii neskol’kikh nauk i nauchnykh napravlenii [hybrid intelligent system of express diagnostics of organizational stress, depression, deviant behavior and anxiety of violators based on convergence of several sciences and scientific directions],” in *Trudy kongressa po intellektual’nym sistemam i informatsionnym tekhnologiyam «IS&IT’17». Nauchnoe izdanie v 3-kh tomakh. [Works of congress on intelligent 17 scientific publication in 3 volumes]*, ser. T. 1. Stupin A. S. publishing House, Taganrog, 2017, pp. 323–329.

[10] A. Palagin, “Problemy transdisciplinarnosti i rol’ informatiki [problems of transdisciplinarity and the role of informatics],” *Kibernetika i sistemnyj analiz [Cybernetics and Systems Analysis]*, no. 5, p. 3–13, 2013.

[11] A. Iliadis, “The tower of babel problem: Making data make sense with basic formal ontology,” 02 2019.

[12] I. Barinov, , N. Borgest, S. Borovik, O. Granichin, S. Grachev, Y. Gromyko, R. Doronin, S. Zinchenko, A. Ivanov, V. Kizeev, R. Kutlakhmetov, V. Laryukhin, S. Levashkin, A. Mochalkin, M. Panteleev, S. Popov, E. Sevastyanov, P. Skobelev, A. Chernyavsky, V. Shishkin, and S. Shlyaev, “Development strategy formation of the committee on artificial intelligence in the scientific and educational center "engineering of the future",” *Ontology of Designing*, vol. 11, no. 3, pp. 260–293, Sep. 2021. [Online]. Available: https://doi.org/10.18287/2223-9537-2021-11-3-260-293

[13] V. Golenkov, N. Guliakina, and D. Shunkevich, *Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

Интеллектуальные компьютерные системы нового поколения и технология комплексной поддержки их жизненного цикла

Голенков В. В., Гулякина Н. А.

В работе рассмотрены принципы построения интеллектуальных компьютерных систем нового поколения, а также принципы построения комплексной технологии их разработки и поддержки жизненного цикла – Технологии OSTIS. В качестве ключевых свойств интеллектуальных систем нового поколения выделяются их семантическая совместимость и интероперабельность. В работе рассматривается подход к обеспечению указанных свойств, реализуемый в рамках Технологии OSTIS.

Received 01.11.2022

General-purpose semantic representation language and semantic space*

Valerian Ivashenko
Department of Intelligent Information Technologies
Belarusion State University of Informatics and Radioelectronics
Minsk, Republic of Belarus
ivashenko@bsuir.by

Abstract—**In the article, models and tools that provide a unified representation of knowledge and their integration within a "semantic space" are considered. For this, the concept of a "generalized formal language" is introduced, which makes it possible to identify the relation between formal languages and known knowledge representation languages such as semantic networks for the purpose of analysis. Based on this analysis, the semantics of the languages of the unified semantic knowledge representation model is specified. A general-purpose language is also introduced as the basis for the technology of developing intelligent systems. And as a result, the concept of "semantic space" is given. The latter is focused on the usage in order to assess the quality of intelligent computer systems within the OSTIS technology. Based on the proposed models, applied problems and further prospects for the development of technologies and their components are considered.**

Index Terms—**Semantic Space, Distensible Sets, Generalized Formal Language, Generalized String, Generalized Kleene's Closure, Set Ordination, Individual Set, Ordered Set, Unified Knowledge Representation Model, Knowledge Specification Model, OSTIS, Knowledge Integration, Introscalar product, Introscalar basis, Taxonomy optimization, Homogeneous Semantic Network, denotational semantics, operational semantics, game semantics, Holomovement, Interoperability, Convergence, Space-Time, Topological Space, Semantic Metric Space, Semantic Metric, Manifold, Becoming, Finite Structure**

I. Introduction

In the article, models and tools that provide a unified representation of knowledge and their integration within the "semantic space" in order to develop a standard for the technology of designing intelligent computer systems are considered [5]. This takes into account various aspects of integration, including the integration of data and knowledge representation levels, as well as dynamic aspects of integration that are closely related to the operational semantics of knowledge [9], [11]. As for the review of languages and models focused on a unified semantic representation of knowledge and their integration, and the corresponding approaches to solving these problems, it is proposed to refer to the work [10] for access to review materials. In this paper, the review part is dedicated to the issues and history of refining the concept of "semantic space". The need to consider knowledge representation languages that provide a unified semantic representation of knowledge is conditioned be the need to represent elements of the semantic space.

In the article, the following will be considered: 1) the formulation of the problem of forming concepts that can express the meaning of the term "semantic space", the formulation of the identified problems that need to be overcome in order to solve the problem; generalization of the concept of formal language [20], and mathematical foundations for the model representation of texts of knowledge representation languages in the form of texts of generalized formal languages; 2) identifying requirements for the alphabet of a knowledge representation language focused on the semantic (meaningful) representation of knowledge, which are the rationale for choosing such a language and identifying such a language among generalized formal languages, as well as languages of the unified semantic knowledge representation model; 3) the language with its core proposed on the basis of the identified abstractions, which provide a unified semantic representation of knowledge in intelligent systems within an open technology for the development of intelligent computer systems; 4) historically formed approaches to the genesis of the concept of "semantic space" and those close to it, including some abstractions of "space" in mathematics; 5) proposed approaches, formalisms, and models for the becoming of concepts capable of expressing the meaning of the term "semantic space" in accordance with the model of a unified semantic representation of knowledge; 6) application of some of the proposed problem-solving models of the level of knowledge control in ontologies on the example of taxonomies; 7) a conclusion containing the main results and prospects for the application of the proposed models.

The semantic space implies the inclusion of various meanings, therefore, an important problem on the way to learning the corresponding "semantic space" concept is the integration of knowledge and the represented meanings.

In knowledge-based systems [22], four directions of integration (of knowledge and models of their representation) are distinguished:

- vertical (introspection);
- horizontal profile (knowledge engineering);
- horizontal frontal (unification);

* The author is grateful to Dr. Tech. Sc., Professor V.V. Golenkov for clarifying the formulation of the problem and the topic of this work, as well as for providing additional materials to the OSTIS standard. The author is also grateful to E. Bobyor, A. Shichko and D.A. Ivashenko for technical assistance and support in the work on the article.

- the direction of continuous integration (training and adaptation).

Problems of integrating knowledge into a single semantic space (unification) are:

- the availability and usage of non-finite structures in models and methods for the representation and processing of knowledge and the formalization of meanings, which make it difficult or exclude the algorithmization of working with such representations, including their analysis and unification;
- uncertainty of models reflecting the results of vertical integration of models and information representation languages, including formal languages, and providing consistent (continuous) integration of texts of dynamic knowledge representation models;
- the presence of different types of knowledge semantics: game [23], [24], [9], operational [9], [25], denotational [5], [26], model [8], etc., requiring correlation in order to identify the equivalence of knowledge;
- dynamic, non-monotonic nature of knowledge, the presence of non-factors [1] in knowledge, the presence of reliable knowledge and knowledge about the unreliable and hypothetical.

Let us formulate the main problems:

- search, comparison, and grounding of the chosen means for knowledge representation;
- search for models for representing and analyzing the structure of the text elements in the languages of the selected knowledge representation model;
- application of the results of the analysis to solve the problems of the level of knowledge operation.

To solve the problem and overcome the identified problems, it is proposed to use methods of mathematical modeling, including embedding (isomorphic injective mapping) of some mathematical structures into others, application of models and methods of theories of formal languages and formal systems, combinatorics, and discrete optimization.

II. Generalization of formal languages and semantic network knowledge representation models

As it is known, the language as a "sign system" is designed to perform "communicative", "cognitive" ("epistemological"), "representative", and other "functions" [27]. Within the language, signs are organized into texts, which have a sequential structure performing a "communicative function": sounds, words of oral speech are organized into a sequence, just like letters, written signs are into lines. Mathematically, a string is an "oriented" ("ordered") "set" (of characters) [29]. The "set" itself is a mathematical abstraction of thinking reflecting its ability to generalize and move from parts to the whole. The question of correlating "oriented sets" and "unoriented sets" belongs to the foundation of mathematics [28]. This question is retaining its actuality. The need to consider this issue is related to the combination of classes "oriented" and "unoriented sets" used in the representation of knowledge, and corresponding to them within the "semantic space" ("semantic metric"), in which it is necessary to correlate elements of these classes (Fig. 1 and 2). There are known particular solutions to this issue for the concept of "oriented" ("ordered") "pairs" proposed by N. Wiener, F. Hausdorff, K. Kuratovsky, and others.

Definition according to N. Wiener [30]:

$$\langle \chi, \gamma \rangle_W \overset{def}{=} \{\{\{\chi\}, \emptyset\}, \{\{\gamma\}\}\}$$

Definition according to F. Hausdorff [30]:

$$\langle \chi, \gamma \rangle_{12} \overset{def}{=} \{\{\chi\} \cup \{1\}\} \cup \{\{\gamma\} \cup \{2\}\}$$

The disadvantage is that either $(\{\chi\} \cap \{2\}) \times (\{\gamma\} \cap \{1\}) = \emptyset$ or $\langle 2, 1 \rangle_{12} = \{\{1, 2\}\}$. In addition, there are lacks for the technical implementation.

Definition according to K. Kuratovsky [31]:

$$\langle \chi, \gamma \rangle_K \overset{def}{=} \{\{\chi\}\} \cup \{\{\chi\} \cup \{\gamma\}\}$$

The disadvantage is that $\langle \chi, \chi \rangle_K = \{\{\chi\}\}$. There are also lacks for the technical implementation.

Other definitions:

$$\langle \chi, \gamma \rangle_{reverse} \overset{def}{=} \{\{\chi\} \cup \{\gamma\}\} \cup \{\{\gamma\}\}$$

Disadvantages are similar to $\langle \chi, \gamma \rangle_K$.

$$\langle \chi, \gamma \rangle_{short} \overset{def}{=} \{\chi\} \cup \{\{\chi\} \cup \{\gamma\}\}$$

The disadvantage is that either $(\{\chi\} \cap \{\{\gamma\}\}) \times (\{\gamma\} \cap \{\chi\}) = \emptyset$ (axiom of foundation) or $\langle \{\gamma\}, \chi \rangle_{short} = \{\chi\}$. It is also a disadvantage from the point of view of type theories that the elements in the set will have a different type, whereas χ and γ are of the same type. For ordinal numbers constructed according to von Neumann, [33] we have $\langle \emptyset, \emptyset \rangle_{short} = \langle 0_{Ord}, 0_{Ord} \rangle_{short} = 2_{Ord}$.

$$\langle \chi, \gamma \rangle_{01} \overset{def}{=} \{\{\chi\} \cup \{0\}, \{\gamma\} \cup \{1\}\}$$

Disadvantages are similar to $\langle \chi, \gamma \rangle_{12}$.

Let
$s(\chi) \overset{def}{=} \{\emptyset\} \cup \{\{\tau\} | \tau \in \chi\}$ then, according to M. Morse, an oriented pair, triple, etc. will be [32]:
$\langle \chi, \gamma \rangle_M \overset{def}{=} (\{0\} \times s(\chi)) \cup (\{1\} \times s(\gamma))$,
$\langle \chi, \gamma, \zeta \rangle_M \overset{def}{=} (\{0\} \times s(\chi)) \cup (\{1\} \times s(\gamma)) \cup (\{2\} \times s(\zeta))$.

The disadvantage is that the Cartesian product uses a pair in accordance with the definition of K. Kuratovsky.

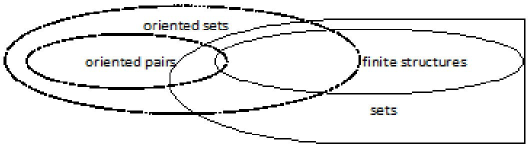


Fig. 1. A correlation diagram for the set and oriented set concepts.

As shown, the above proposals do not solve the issue in general or have their drawbacks. Other definitions rely on the existence of infinite structures (Fig. 1).

Let us define an oriented set σ in the following way :
$\sigma \overset{def}{=} \bigcup_{\iota=1}^{|\sigma|} \left\{ (|\sigma| - \iota + 1)^{\{\sigma_\iota\}} \right\}_\iota$.

An oriented set σ is the combination of individual sets of order ι of all ordinations of its components singletons σ_ι to the base of $|\sigma| - \iota + 1$, where ι takes values from 1 to $|\sigma|$.

Ordination of the set σ to the base 1:
$1^\sigma \overset{def}{=} \sigma$.

Ordination of the set σ to the base $\iota + 1$:
$(\iota + 1)^\sigma \overset{def}{=} \{\iota^\tau \, | \tau \subseteq \sigma\}$.

It should be noted that the logarithm of a set σ to the base 2 is its boolean 2^σ [34] .

An individual set of order 1 of element χ:
$\{\chi\}_1 \overset{def}{=} \{\chi\}$.

An individual set of order $\iota + 1$ of element χ:
$\{\chi\}_{\iota+1} \overset{def}{=} \{\{\chi\}\}_\iota$.

It should be noted that an oriented set of one element is :
$\langle\chi\rangle \overset{def}{=} \left\{1^{\{\chi\}}\right\}_1 = \{\{\chi\}\}$, an oriented set of two elements coincides with an oriented pair according to N. Wiener :
$\langle\chi, \gamma\rangle \overset{def}{=} \left\{2^{\{\chi\}}\right\}_1 \cup \left\{1^{\{\gamma\}}\right\}_2 = \{\{\{\chi\}, \emptyset\}\} \cup \{\{\{\gamma\}\}\} = \{\{\{\chi\}, \emptyset\}, \{\{\gamma\}\}\}$, and the oriented triple is :
$\langle\chi, \gamma, \zeta\rangle \overset{def}{=} \left\{3^{\{\chi\}}\right\}_1 \cup \left\{2^{\{\gamma\}}\right\}_2 \cup \left\{1^{\{\zeta\}}\right\}_3 = \{\{\{\{\chi\}, \emptyset\}, \{\emptyset\}\}, \{\{\{\gamma\}, \emptyset\}\}, \{\{\{\zeta\}\}\}\}$.

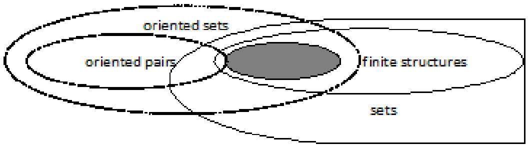


Fig. 2. A correlation diagram for the set and defined oriented set concepts.

Thus, according to this definition, oriented sets are sets and finite structures (Fig. 2).

If it is necessary for some non-empty oriented sets to be non-oriented, it is possible to define an own subclass as unoriented sets, according to the definitions:
$\bigcup_{\tau\in\cdot\chi} \{[\tau]\} \overset{def}{=} \bigcup_{\tau\in\cdot\chi} \left\{2^{\{\tau\}}\right\}_{\kappa(\chi)}$; $\chi \overset{def}{=} \bigcup_{\tau\in\cdot\chi} \left\{2^{\{\tau\}}\right\}_{\kappa(\chi)}$,
where $\kappa(\chi) \overset{def}{=} 1$.

For example:
$\{[\chi]\} \cup \{[\gamma]\} \overset{def}{=} \{\{\{\chi\}, \emptyset\}\} \cup \{\{\{\gamma\}, \emptyset\}\}$.

Knowledge representation languages (formal languages) and semantic networks with a graph structure are used to represent knowledge, but there is no known general model in which these knowledge representation means can be compared in order grounding the choice of anyone of them.

Let us extend a class of languages beyond the known class of formal languages in order to ground the choice of the means (language) of knowledge representation to overcome the problem of the lack of known models that reflect the results of vertical integration [11] of models and information representation languages and provide consistent (continuous) integration [11] of texts of dynamic knowledge representation models (Fig. 4).

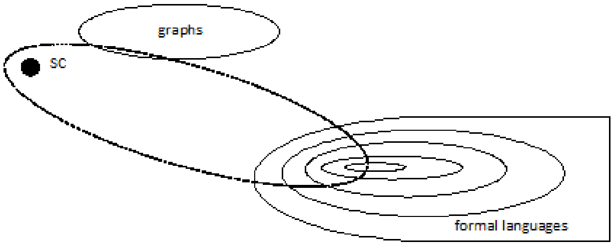


Fig. 3. The correlation of syntax knowledge representation means.

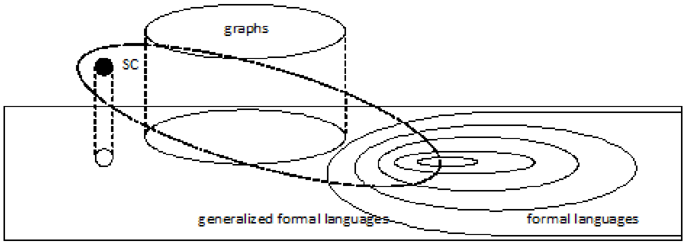


Fig. 4. The correlation of generalized formal languages and knowledge representation means.

We introduce the concept of generalized Kleene closure and generalized formal language [3], [20] in order to extend the class of languages beyond formal languages. The expediency of this extension is conditioned by the need to endow knowledge representation languages with associative properties. The associative properties of a language are reduced to the presence of associations in its texts. The simplest associations are abstract connections (connectives), which are considered mathematically as sets (or directed sets). We will also consider only oriented sets as associations (strings and generalized strings, i.e. strings whose components can only be symbols of the alphabet of the language, strings from them and other generalized strings) in order for the texts of the language to remain finite and the “communicative function” of the language to be preserved.

As it is known, a formal language Λ is a subset of the Kleene closure [20] of its alphabet. A:
$\Lambda \subseteq \mathrm{A}^* / \left(\mathrm{A}/\mathrm{A}^1\right)$,
where the Kleene closure of the alphabet A (the definition is slightly modified to preserve the extensiveness and idempotency of the closure operator):
$\mathrm{A}^{*\Sigma} \overset{def}{=} \mathrm{A} \cup \left(\bigcup_{\iota\in\mathbf{N}/\{0\}} \mathrm{A}^\iota\right)^{\oplus\Sigma}$.

$$\mathrm{A}^* \overset{def}{=} \mathrm{A}^{*\mathrm{A}}$$

The operation $\oplus_\mathrm{A}$ for each element χ of the set to which it is applied, if γ is a string, “insert” in order all the components of the string χ, which is a component of the string χ, into this string χ instead of (γ) if each inserted component belongs to A and the string γ itself does not belong to A. Empty lines are excluded from the string unless they are in A.

Here and below:

- $A \times B$ is the Cartesian product of set A and set B;

- A^n is the Cartesian power n of set A;
- B^A is the exponential (a set of completely defined functions with domain A and domain B);
- B_+^A is the extended exponential (set of functions with origin A and range B ($B^A \subseteq B_+^A$));
- 2^A is the boolean of set A (set of all subsets of set A ($B_+^A \subseteq 2^{A \times B}$)) ;
- R^{-1} is the inverse binary relation to R;
- $R \circ S$ is the composition of binary relations R and S;
- R° is the transitive closure of the binary relation R.

A generalized formal language Λ is a subset of the generalized Kleene closure of the alphabet A:

$$\Lambda \subseteq \mathrm{A}^{(*^*)} / \left(\mathrm{A}/\mathrm{A}^1\right).$$

The generalized Kleene closure of the alphabet A satisfies the following definitions:

$$\mathrm{A}^{(*^*)} \overset{def}{=} \bigcup\nolimits_{\iota \in \mathbf{N}/\{0\}} \mathrm{A}^{(*^\iota)};$$

$$\mathrm{A}^{\left(*^{\iota+1}\right)} \overset{def}{=} \left(\mathrm{A}^{(*^\iota)} \cup \mathrm{A}\right)^*;$$

$$\mathrm{A}^{\left(*^1\right)} \overset{def}{=} \mathrm{A}^*.$$

The generalized Kleene closure is extensive:

$$\mathrm{A}^{(*^*)} \cup \mathrm{A} = \mathrm{A}^{(*^*)}.$$

The generalized Kleene closure is monotonic:

$$\mathrm{A}^{(*^*)} \subseteq (\mathrm{A} \cup \Delta)^{(*^*)}.$$

The generalized Kleene closure is idempotent:

$$\left(\mathrm{A}^{(*^*)}\right)^{(*^*)} = \mathrm{A}^{(*^*)}.$$

The cardinalities of the generalized Kleene closure and the Kleene closure of the non-empty either finite or countable alphabet A are equal (aleph-0).

A generalized formal language Λ is called a symmetric language [3] if and only if for any Δ:

$$((\emptyset \subset \Delta^n \cap \Lambda) \to (\Delta^n \subseteq \Lambda)).$$

Symmetric languages, as a rule, correspond to languages in which the order of transmission of text elements is not essential for the preservation of the transmitted meaning. Among such languages, there are languages focused more on the performance of "cognitive" and "representative" "functions" rather than "communicative", i.e. refer to representation languages.

A generalized formal language Λ is called an associative language [3] if and only if:

$$\exists \mathrm{T} \exists \Delta \exists n \left(\emptyset \subset (\Delta^n \cup \mathrm{T})^* \cap \Lambda\right).$$

A generalized formal language Λ is called a pseudograph (graph) language if and only if A exists and for any Δ, T, n:

$$\Lambda \subseteq \left(\mathrm{A}^2 \cup \mathrm{A}\right)^*;$$

$$\left(\left(\emptyset \subset \left(\Delta^2 \cup \mathrm{T}\right)^n \cap \Lambda\right) \to (\Delta \subseteq \mathrm{T})\right).$$

A syntactically distinguishable fragment of text is a fragment of text (subtext) whose structure differs from others or its position in such language texts as (semantically equivalent) permutations of elements of one of them differs from other fragments (is unique) relative to the structure of all such texts [8].

The consideration of generalized formal languages as a model for knowledge representation languages is conditioned by:

- the finite structure of texts;
- the ability to consider knowledge representation means with a more complex structure of associations than in graphs, in particular, the ability to construct texts that one-to-one correspond to abstract simplicial complexes [3], [35];
- the possibility of associative coding of meaning (representation of knowledge) by associations of a given level and below within the hierarchical structure of the text, which allows for continuous integration (monotonicity) of knowledge represented in this way, changing only the structure of connections in associations of upper levels (in the limit, only the order of elements in the text) but not the state of text elements during processing information in dynamic (procedural, non-monotonic) models of knowledge processing;
- the possibility of syntactic (hierarchical) inclusion of texts of one language into the texts of other languages as a whole (without including their parts and without reflecting their structure) which allows working with texts of several languages from the position of a single model in a single syntactic space and ensure representing the results of processing texts of one language in another, which is typical for vertical integration processes [11];
- the natural way of the syntax reflection of such artificial intelligence languages as LISP [36], [8].

Statement. There are $p * (p+1)/2$ connected distinguishable text fragments for texts with p characters with a linear association structure [8].

Statement. There are $\left(\left(\left(\sqrt{5}+1\right)/2\right)^{2*p+1} - \left(\left(1-\sqrt{5}\right)/2\right)^{2*p+1}\right)/\sqrt{5} - 1$ distinguishable text fragments for texts with p characters with a linear association structure [8].

Statement. There are at least $(\lceil (q-p)/(2*p) \rceil + p - 1)!/(p! * (\lceil (q-p)/(2*p) \rceil - 1)!)$ connected distinguishable text fragments for connected texts with p symbols with a non-linear structure of q ($q \geq p$) associations [8].

The choice of pseudograph (graph) languages is determined by (Fig. 5):

- the ability to build a non-linear association structure, which makes it possible to achieve a number (exponentially dependent on the size of the text) of connected syntactically distinguishable text fragments which are potential answers to questions to the knowledge base, in contrast to languages with a linear association structure (corresponding to formal languages that are not associative), which have

only a quadratic number of connected syntactically distinguishable fragments of texts [8];
- the fact that hypergraph languages do not qualitatively raise the number of connected syntactically distinguishable fragments in their texts but cause difficulties in implementation.

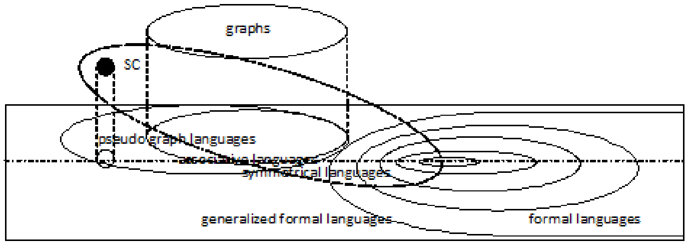


Fig. 5. Symmetrical, associative, and pseudograph languages.

III. Unified representation

Requirements for the knowledge representation language [5]:

- semantic representation, each element of the language text, with the exception, perhaps, of syntactic connectives of their incidence, should be a designation (sign) of the entity;
- universality of representation:
  - representation of phenomena of arbitrary structure, including associations of an unlimited number of elements,
  - representation of knowledge and expression of semantics of any kind;
- simple syntactic structure of texts;
- minimum number of alphabet elements types, unification of representation;
- ensuring the basic properties of knowledge representation (the possibility of eliminating synonymy and bringing it to homogeneous (refined) semantic networks).

A simple syntactic structure of texts means a case when the number of incident connectives is expressed as no more than a linear dependence on the number of symbols in the text, i.e. the relation of the number of incident connectives to the number of symbols in the text does not exceed a certain constant. So, for example, if there are m nodes and n arcs in the text, each of which has three connectives of incidence, the dependence has the following form $3 * n$, and the relation is expressed by the formula $\frac{3*n}{n+m}$, what is less than three for any natural m and n. In the case of a complex syntactic structure, the number of incident connectives in the texts of pseudograph languages can be expressed as a dependence on the number of designations in the text, reaching n^2.

Thus, the alphabet close to the minimum one I should contain:

- designations of entities (including connectives) that can have an unlimited number of incident designations (and corresponding incident connectives);
- designations (of basic connectives), which can have a limited number of incident designations (and corresponding incident connectives), in the simplest case — equal to two;
- designations (of common connectives) that can have a limited number of incident designations (and corresponding incident connectives), in the simplest case — equal to two.

Table I
Comparison of languages with different types of alphabet elements

	one kind			***two kinds***				***more than two kinds***
syntactic restrictions	-	+		-	+			+
simple syntactic structure	-	+		-	+			+
semantic representation	+	+	-	+	+	-	+	+
arbitrary phenomena structure representation	+	-	+	+	-	+	+	+
ability to represent any knowledge and to express semantics of any kinds	-			+			-	+

Basic and common connectives are assumed to be oriented, since unoriented connectives are relatively easy to define with a pair of oriented connectives and the designation of an entity that does not denote a basic or common connective. In order to define an oriented connective through unoriented connectives, at least three unoriented connectives and at least two entity designations are required that do not denote the basic or common unoriented connectives.

Thus, the alphabet of the required knowledge representation language (core) must specify designations of at least three types: vertices ($V_{E_{SC}}$) and two types of arcs ($B_{E_{SC}}$, $C_{E_{SC}}$):

$$E_{SC} = V_{E_{SC}} \cup B_{E_{SC}} \cup C_{E_{SC}}.$$

Each designation may belong to (be a member of) more than one type, however, for each occurrence of the designation in the text of the required knowledge representation language, of which this designation is an element, its type is determined unambiguously.

For arcs, it is possible to define a single view as the union of both types of arcs:

$$E_{E_{SC}} = B_{E_{SC}} \cup C_{E_{SC}}.$$

If we consider the designations of the uncertain type $U_{E_{SC}} \subseteq E_{SC}$, convenient when representing knowledge in the presence of such NON-factors [1] as incompleteness, then the remaining types of designations can be expressed as:

$$V_{E_{SC}} = U_{E_{SC}} / E_{E_{SC}},$$

$$C_{E_{SC}} = E_{E_{SC}} / B_{E_{SC}}.$$

The alphabet of the required language of knowledge representation contains designations of the types:

- elements of the uncertian type $U_{E_{SC}}$;
- arcs of an unified type $E_{E_{SC}}$;
- basic arcs $B_{E_{SC}}$.

The alphabet of the extension of the required language for knowledge representation differs in the contents of a larger number of types of designations (elements):

- elements of an uncertain type (vertexes) $U_{E_{SC}}$;
- nodes $V_{E_{SC}}$;
- arcs of permanent membership $P_{E_{SC}}$;
- arcs of temporal actual membership (basic arcs) $A_{E_{SC}}$ ($B_{E_{SC}}$);
- arcs of temporal phantom membership $T_{E_{SC}}$;
- arcs of fuzzy membership (arcs of an unified type) $F_{E_{SC}}$ ($E_{E_{SC}}$);
- arcs of temporal phantom non-membership $H_{E_{SC}}$;
- arcs of temporal actual non-membership $G_{E_{SC}}$;
- arcs of permanent non-membership $N_{E_{SC}}$.

The listed types of elements are connected by the following relations:
$U_{E_{SC}} \cup V_{E_{SC}} \cup E_{E_{SC}} \subseteq E_{SC}$, where
$P_{E_{SC}} \cup A_{E_{SC}} \cup T_{E_{SC}} \cup F_{E_{SC}} \cup H_{E_{SC}} \cup G_{E_{SC}} \cup N_{E_{SC}} = E_{E_{SC}}$,

- a node cannot be an arc: $V_{E_{SC}} \cap E_{E_{SC}} = \emptyset$;
- the arc of permanent membership cannot be the arc of temporal membership, the arc of permanent membership cannot be the arc of temporal non-membership, the arc of permanent non-membership cannot be the arc of temporal membership, the arc of permanent non-membership cannot be the arc of temporal non-membership: $(P_{E_{SC}} \cup N_{E_{SC}}) \cap (A_{E_{SC}} \cup T_{E_{SC}} \cup H_{E_{SC}} \cup G_{E_{SC}}) = \emptyset$;
- the arc of permanent non-membership cannot be the arc of permanent membership: $P_{E_{SC}} \cap N_{E_{SC}} = \emptyset$.

Such types of elements and their relations are conditioned by the need to solve the problems of horizontal profile integration [11] by unifying the representation.

When there is mapping into generalized formal languages for the purpose of vertical integration of languages (texts) for representation and reduction of texts to the fundamental alphabet A_{SC}, the following is true:

$$E_{SC} \tilde{\subset} {A_{SC}}^{(*^*)}$$

$${E_{SC}}^2 \tilde{\subset} {A_{SC}}^{(*^*)}.$$

The model of the unified semantic representation of knowledge [8] is set by a triple:

$$\langle S_{SC}, R_{SC}, F_{SC} \rangle$$

The model languages of the unified semantic representation of knowledge are defined based on the elements of the alphabet and syntax, the set of all texts forms a general sc-language (Semantic Code, L_{SC} language), as well as the set of all its subsets ($S_{SC} = 2^{L_{SC}}$) is sc-languages (sc-sublanguages):

$$L_{SC} \tilde{\subset} {A_{SC}}^{(*^*)}.$$

The syntax of the model languages for the unified semantic representation of knowledge (sc-languages) [8] describes the properties of connectives of incidence relations in the alphabet elements of these languages in their texts. Two incidence relations for elements of the alphabet of these languages are distinguished I_{SC} and C_{SC}:

$$I_{SC} \tilde{\subseteq} {E_{SC}}^2;$$

$$C_{SC} \subseteq I_{SC}.$$

Based on these relations I_{SC}, C_{SC} one incidence relation R can be determined:

$$R = (I_{SC} - C_{SC})^{-1} \cup I_{SC};$$

$$R \tilde{\subseteq} {E_{SC}}^2;$$

$$L_{SC} \tilde{\subseteq} (R \cup E_{SC})^*.$$

The syntax of sc-languages can be set as follows. For each text S of sc-languages and the set of all its components and only them, X ($S \tilde{\in} X^n$):

$$(\forall Y \left((Y \subset X) \rightarrow (\neg \left(S \tilde{\in} Y^n \right)) \right)),$$

there will be such sets of incident connectives I, C and the set of occurrences of designations T (terminals), V (nodes), E (edges), A (arcs), and B (basic arcs), that:

$$|I \cup C| + |T \cup V \cup E| = n; I \tilde{=} I_{SC} \cap X; C \tilde{=} C_{SC} \cap X;$$

$$A \tilde{=} E_{E_{SC}} \cap X; B \tilde{\subseteq} (P_{E_{SC}} \cup A_{E_{SC}}) \cap X; T \cup V \cup E \tilde{\subseteq} X;$$

$$V \tilde{\subseteq} V_{E_{SC}} \cap X; I \cap C = C; V \cap A = \emptyset; E \cap A = A; A \cap B = B,$$

neither terminals nor nodes that are not edges are incident to each other:

$$I \cap ((T \cup V/E) \times (T \cup V/E)) = \emptyset;$$

any edge is incident to at least one designation:

$$\forall e \left((e \in E) \rightarrow (\emptyset \subset I \cap (\{e\} \times (V \cup T \cup E))) \right);$$

no more than two designations are incident to any edge:

$$\forall e \left((e \in E) \rightarrow (|I \cap (\{e\} \times (V \cup T \cup E))| \leq 2) \right);$$

one (second) designation is incident to any arc:

$$\forall e \left((e \in A) \rightarrow (|C \cap (\{e\} \times (V \cup T \cup E))| = 1) \right);$$

at least one node is incident to any edge, if it is a basic arc, the element that is not incident to it (not the second one) is a node:

$$\forall e ((e \in B) \rightarrow ((\emptyset \subset I \cap (\{e\} \times V)) \wedge ((I - C) \\ \cap (\{e\} \times V) = (I - C) \cap (\{e\} \times (V \cup T \cup E))))).$$

There are interpretations of the texts of model languages for the unified semantic representation of knowledge as texts of the symmetric (pseudo-) graph (or multipseudograph) language (see Fig. 6, Fig. 7, and Fig. 8). Where $k + 1$ order associations correspond to connectives of incidence relations and sc-elements (designations in the texts of model languages for the unified semantic knowledge representation) correspond to k order associations of the fundamental alphabet A_{SC}.

When representing connectives of both incidence relations in the texts of a generalized formal language, duplication of connectives is allowed (multipseudograph), the second occurrence of the connective corresponds to belonging to the second relation, each designation corresponds to the vertex of the multipseudograph (Fig. 6):

$$\langle\langle a,b\rangle, \langle a,e\rangle, a, b, e, \langle a,e\rangle\rangle.$$

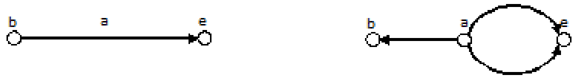


Рис. 6. Multipseudograph representation (right) of SC-text (left)

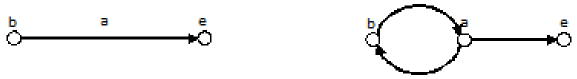


Fig. 7. Pseudograph representation (right) of SC-text (left)

Transformation of arcs of a unified type (sc-arcs), to which the nodes (sc-nodes) are incident (Fig. 7).

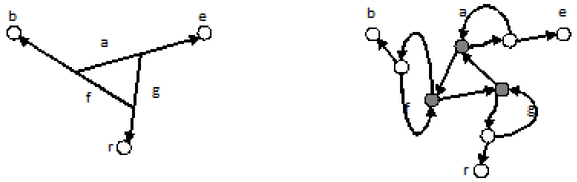


Fig. 8. Pseudograph representation (right) of SC-text with incident arcs (left)

Transformation of arcs of a unified type (sc-arcs), to which arcs of a unified type (sc-arcs) are incident (Fig. 8).

Transition from text to oriented pseudograph (Fig. 7 and Fig. 8):

- mapping the vertices of elements (sc-elements);
- mapping of arcs to connections (connectives) of incidence.

To reduce “changes in state” of “memory elements” to “changes in the connections between them” when representing texts of sublanguages of symmetric associative (pseudo)graph (or multipseudograph) language, it can be agreed that the actual elements (sc-elements, sc-arcs) are separated from the phantom elements (sc-elements, sc-arcs) by connectives of incidence relations (some are before those, and others are after).

Representation for connectives of relations of actual and phantom membership (Fig. 9) in the texts of a generalized formal language:

$$\langle\langle a,b\rangle, \langle a,e\rangle, \langle f,r\rangle, \langle f,a\rangle, a, b, e, r, \langle a,e\rangle, \langle f,a\rangle, f\rangle.$$

Relations of the unified semantic knowledge representation model [8]:

- sublanguage (sc-sublanguage),
- injective language mapping (of sc-languages).

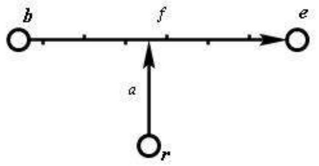


Fig. 9. SC-language text with connectives of actual and phantom membership

Features (functions) of the model languages of the unified semantic representation of knowledge (sc-languages) [8]:

- proper and non-proper sublanguage key elements for (pairs of) languages (language and sublanguage);
- semantic neighborhoods of key elements in sublanguage texts for language pairs;
- semantic interpretations of text elements.

When representing arcs of a unified type (sc-arcs), it can be assumed that they denote a pair $\langle\langle x,\mho\rangle, y\rangle$.

IV. Internal language of ostis-systems

The internal language of ostis-systems – an ***SC-code*** (Semantic Computer code) – is a model language of unified semantic representation, that is, the language of unified semantic representation of knowledge in the memory of intelligent computer systems [5].

An ***SC-code*** is [5]:

- an abstract language, that is, a language for which the way of representing symbols (syntactically elementary fragments) that are part of the texts of this language is not specified, but only the alphabet of these symbols is specified, that is, a family of character classes that are considered syntactically equivalent to each other;
- a pseudograph language;
- a universal language that provides internal representation and structuring of all (!) knowledge used by the ostis-system in the course of its functioning, and is the result of unification (refinement) of syntax and denotation semantics of semantic networks.

Graph language (pseudograph language) is a language, each text of which [5]:

- is defined by a set of elementary fragments (symbols) included in it, which, in turn, consists of a set of nodes (vertices), possibly of syntactically different types, and a set of connective designations, which may also belong to different syntactically distinguished classes;
- is defined in the general case by several relations of the incidence of connective designations with the components of these connectives (in this case, the specified components in the general case can be not only nodes but also connective designations).

Each abstract language can be matched with a family of real languages that provide an isomorphic real representation of the texts of the specified abstract language by clarifying the ways of representation (images, encoding) of the symbols included in these texts, as well as by clarifying the rules for establishing syntactic equivalence of these symbols. In all other respects, the syntax and denotational semantics of these real languages are completely similar and correspond to the syntax and denotational semantics of the corresponding abstract language [5].

The universality of the ***SC-code*** is also ensured by the fact that the elements of sets denoted by the elements of ***SC-code*** texts can be signs of the described entities of any kind, including signs of connections between the described entities and/or their signs [5].

Texts of the ***SC-code*** are graph structures of an extended form, in which the designations of the described connections can connect not only the vertices (nodes) of the graph structure but also the designations of other connections [5].

The ***SC-code*** is the basic universal language of the internal representation of knowledge in the ostis-systems memory (the basic internal language of ostis-systems) [5], this means that it is the maximum internal language of ostis-systems, in relation to which all other (specialized) internal languages are its sublanguages (subsets), that is, it is a set of all possible text of the ***SC-code*** (sc-texts). The signs (designations) of all entities described in sc-texts (texts of the ***SC-code***) are represented as syntactically elementary (atomic) fragments of sc-texts and, therefore, do not have an internal structure in the same sc-text, not consisting of simpler fragments of sc-text, such as names (terms), which represent the signs of the described entities in familiar languages and consist of letters. Names (terms), natural language texts, and other information constructions (generalized strings) that are not sc-texts can be contained in sc-elements included in the sc-text as files described (specified) by sc-texts [5], [3]. Thus, the knowledge base of an intelligent computer system built on the basis of the ***SC-code*** may contain names (terms) denoting some of the described entities in the form of corresponding files. Each sc-element will be called an internal designation of some entity, and the name of this entity, in the form of a file (sc-file), will be called an external identifier (external designation) of this entity. An external identifier can be not only a name (term) but also a hieroglyph, a pictogram, a voiced name, a gesture. It should be particularly noted that the external identifiers of the described entities in an intelligent computer system built on the basis of the ***SC-code*** are used for: (1) analyzing information coming into this system from outside from various sources and entering (understanding and immersing) this information into the knowledge base, (2) synthesis of various messages addressed to various subjects (including users).

Texts of the ***SC-code*** (sc-texts) generally have a pseudograph (graph, nonlinear) structure, since the sign of each described entity can be incident to an unlimited number of other signs, since each described entity can be connected by an unlimited number of connections with other described entities [5].

The knowledge base, represented by the text of the ***SC-code***, is a graph structure of a special kind, the alphabet of elements of which includes many elements of the explained type, many nodes, many basic arcs – arcs of a specially highlighted type that provide structuring of knowledge bases, as well as [8] a set of arcs of permanent non-membership, a set of arcs of temporal actual non-membership, a set of arcs of temporary phantom non-membership, a set of arcs of fuzzy membership, a set of arcs of phantom membership, a set of arcs of permanent membership, a set of special nodes, each of which has content that is a file stored in the memory of an intelligent computer system. The structural feature of this graph structure is that its arcs and edges can connect not only a node with a node but also a node with an arc or an arc with another arc [5].

An arc is the designation of a binary oriented connective between two entities. An arc of a special kind (base arc) is a sign of connection between a node denoting a certain set of elements of the graph structure under consideration and one of the elements of this graph structure that currently belongs to the specified set. At the same time, the connections denoted by the elements of the graph structure under consideration can be permanent (always existing) and temporal (connections that correspond to the period of their existence) [5].

In the considered graph structure, which is a representation of the knowledge base in the ***SC-code***, there may but should not exist different elements of the graph structure denoting the same entity. If a pair of such elements is detected, then these elements can be pasted together (equated). Thus, the synonymy of internal designations in the knowledge base of an intelligent computer system built on the basis of the ***SC-code*** is undesirable. At the same time, the synonymy of external designations is considered as a normal phenomenon [5].

In addition to files (***sc-files***) representing various external designations (names, hieroglyphs, pictograms), files of various texts (books, articles, documents) can be stored in the memory of an intelligent computer system built on the basis of the ***SC-code***, notes, comments, explanations, drawings, pictures, schemes, photographs, video and audio materials [5].

Any entity, that is capable of having a designation, in the text of the ***SC-code*** can be associated with an sc-element denoting a set to which only the designation of this entity belongs. This is one of the factors that ensure the universality of the ***SC-code***. We emphasize that sc-elements are not just designations but designations that are elementary (atomic) fragments of a sign construction, i.e. fragments whose detailed structure is generally not required for ”reading” and understanding this sign construction [5].

The text of the ***SC-code***, like any other graph structure, is an abstract mathematical object that does not require detailing (refinement) of its encoding in the memory of a computer system (for example, in the form of an adjacency matrix, an incidence matrix, a list structure). But such detailing will be required for the technical implementation of the memory in which sc-texts are stored and processed [5].

The most important additional property of the ***SC-code*** is that it is convenient not only for the internal representation of knowledge in the memory of an intelligent computer system but also for the internal representation of information in the memory of computers specifically designed to interpret semantic models of intelligent computer systems. That is, the ***SC-code*** defines the syntactic, semantic, and functional principles of organizing the memory of next-generation computers focused on the implementation of intelligent computer systems – the principles of organizing graphodynamic associative semantic memory [5].

The ***SC-code*** includes the ***SC-code Core*** and is considered as an ***Extension of the SC-Code Core*** [5].

It should be emphasized that unification and the maximum possible simplification of syntax and denotational semantics in the internal language of intelligent computer systems are primarily necessary because the overwhelming amount of knowledge stored in the knowledge base of an intelligent computer system are meta-knowledge describing the properties of other knowledge. Meta-knowledge, in particular, should include various kinds of logical propositions and various kinds of programs, descriptions of methods (skills) that provide solutions to various classes of problems. It is necessary to exclude the dependence of the form of the represented knowledge on the type of this knowledge. The form (structure) for the internal representation of knowledge of any kind should depend only on (!) from the meaning of this knowledge [5].

Moreover, constructive (formal) development of the theory of intelligent computer systems is impossible without clarification (unification, standardization) and ensuring semantic compatibility of various knowledge types stored in the knowledge base of an intelligent computer system. It is obvious that the variety of forms for representing semantically equivalent knowledge makes the development of a general theory of intelligent computer systems practically impossible [5].

The ***Alphabet of the SC-code Core*** [5], as well as the alphabet of the required knowledge representation language, contains:

- ***sc-elements of an uncertain type*** (vertexes, ***sc-elements***) $(= U_{E_{SC}})$ [1];
- ***sc-arcs*** (arcs of a unified type) $(= E_{E_{SC}})$;
- ***basic sc-arcs*** (basic arcs) $(= B_{E_{SC}})$.

During processing the text of the ***SC-code Core***, from text to text, the syntactic type of ***sc-elements*** can be specified – an ***sc-element of an uncertain type*** can be an ***sc-arc***, an ***sc-arc – basic sc-arc***.

The ***Alphabet of the SC-code Core*** corresponds to the features of the classification of sc-elements and sets the syntactic classification of sc-elements [5].

Syntactic classification of ***sc-elements*** of the ***SC-code Core*** [5]:

- ***sc-elements of an uncertain type*** $(= U_{E_{SC}})$;
  - ***sc-nodes*** $(= V_{E_{SC}})$;
  - ***sc-arcs*** $(= E_{E_{SC}})$;
    - ***sc-arcs of a common type*** $(= C_{E_{SC}})$;
    - ***basic sc-arc*** $(= B_{E_{SC}})$.

All classes of sc-elements included in the syntactic classification of sc-elements are syntactically highlighted classes of sc-elements [5]. The ***SC-code*** is referred to as the syntactic extension of the ***SC-code Core***, since the ***Alphabet of the SC-code*** is an extension of the ***Alphabet of the SC-Code Core***.

The syntactic extension of the SC-code Core consists in the introduction of an additional class of syntactically equivalent elementary fragments of constructions of the***SC-code Core*** – sc-elements designating internal files stored in the ostis-system memory [5].

All files representing electronic images of information constructions external to the ***SC-code*** can be represented in the ***SC-code*** using graph structures in which sc-elements designate letters of texts or pixels of images [5].

The most important type of internal files of ostis-systems are files of external identifiers of sc-elements (in particular, names of sc elements) representing sc-elements in texts of external languages (including in texts of SCs- and SCn-codes) [8], [10], [5]. The ***Set of all the elements of the SC-code Core constructions*** and the ***Set of all the elements of the SC-code constructions*** completely coincide, since for each element of the ***SC-code Core*** construction there is an element synonymous with the ***SC-code*** construction and vice versa. It follows from this that the semantic classification of the elements of ***SC-code*** and ***SC-code Core*** information constructions are also completely identical. Everything that can be designated and described by texts of the ***SC-code*** can be designated and described by texts of the ***SC-code Core***. The difference between the ***SC-code*** and the ***SC-code Core*** is that a new syntactically highlighted class of sc-elements is added to the ***SC-code*** – a class of sc-elements that are signs of specific (constant) files stored in the ostis-system memory. Such "internal" files are necessary so that information constructions that are not texts of the ***SC-code*** can be stored and processed in the ostis-system memory, which is necessary when entering (perceiving) information coming from outside, as well as when generating information structures transmitted to other subjects. The inclusion in the ***SC-code*** of special syntactically highlighted sc-nodes denoting electronic images (files) of various types of information constructions that are not SC-code constructions makes it possible to process not only in the ostis-system memory, that is, in the same storage environment, not only ***SC-code*** constructions but also constructions "external" for it. Without the implementation of the ostis-system interface, it is impossible to implement syntactic analysis, semantic analysis, and understanding, as well as it is also impossible to realize the synthesis (generation) of external information constructions belonging to a given external language and semantically equivalent to a given meaning. Since all the syntactic and semantic properties of the ***SC-code*** and the ***SC-code Core*** are very close, when describing the ***SC-code***, attention is focused on its differences from the ***SC-code Core***, as well as for a more detailed consideration of the semantic classification of elements [5].

The ***Syntax of the SC-code*** differs from the ***Syntax of the SC-Code Core*** by the fact that in the ***Alphabet of the SC-code***, the class of ***sc-elements*** is additionally introduced, which are signs of files stored in the ostis-system memory [5].

The ***Alphabet of the ostis-systems language***, the Alphabet of sc-elements within the SC-code, the alphabet of the extension

[1] Similarly (by the equality of the name in parentheses) we will denote synonyms of key elements of knowledge representation languages, in order to reduce the length of formulas in which they are used as synonymous names in a local context

of the required knowledge representation language contains:

- ***sc-element of an uncertain type*** (vertexes, ***sc-elements of the common type***) ($= U_{E_{SC}}$);
- ***sc-elements with contents*** (***sc-files***) ($= D_{E_{SC}}$);
- ***sc-nodes*** (nodes) ($= V_{E_{SC}}$);
- ***sc-arcs of permanent membership*** (arcs of the permanent membership) ($= P_{E_{SC}}$);
- ***sc-arcs of temporal actual membership*** (basic arcs, ***basic sc-arcs***) ($= A_{E_{SC}}$ ($B_{E_{SC}}$));
- ***sc-arcs of temporal phantom membership*** (arcs of temporal phantom membership) ($= T_{E_{SC}}$);
- ***sc-arcs of fuzzy membership*** (arcs of fuzzy membershipи, arcs of a unified type ($= F_{E_{SC}}$ ($= E_{E_{SC}}$));
- ***sc-arcs of temporal phantom non-membership*** (arcs of temporal phantom non-membership) ($= H_{E_{SC}}$);
- ***sc-arcs of temporal actual non-membership*** (arcs of temporal actual non-membership) ($= G_{E_{SC}}$);
- ***sc-arcs of permanent non-membership*** (arcs of permanent non-membership) ($= N_{E_{SC}}$).

The ***Alphabet of the ostis-systems language*** is [5]:

- a family of maximum sets of syntactically equivalent (within the ***SC-code***) sc-elements;
- a family of classes with syntactically equivalent sc-elements of the ***SC-code***;
- a family of sets, each of which includes all syntactically equivalent to each other (within the ***SC-code***) sc-elements and only them.

The ***Alphabet of the ostis-systems language*** sets the signs (parameters) of syntactic equivalence of sc-elements [5].

The set of all elements of the ***SC-code*** constructions coincides with the set of all elements of the ***SC-code Core*** constructions. Just in the ***SC-code*** constuctions some sc-elements having a "syntactic label" (syntactic type) of an ***sc-element of a common type***, will have the "label" of the sc-element, which is the sign of an internal file stored in the ostis-system memory [5].

Syntactic classification of sc-elements of the SC-code [5]:

- ***sc-element of an uncertain type*** (vertexes, ***sc-elements***) ($= U_{E_{SC}}$);
  - ***sc-nodes*** (nodes) ($= V_{E_{SC}}$);
  - ***sc-arcs*** (***sc-arcs of fuzzy membership***) ($= F_{E_{SC}}$ ($E_{E_{SC}}$));
    - ***sc-arcs of a common type*** ($= C_{E_{SC}}$);
      - ***sc-arcs of permanent membership*** ($= P_{E_{SC}}$);
      - ***sc-arcs of temporal phantom membership*** ($= T_{E_{SC}}$);
      - ***sc-arcs of temporal phantom non-membership*** ($= H_{E_{SC}}$);
      - ***sc-arcs of temporal actual non-membership*** ($= G_{E_{SC}}$);
      - ***sc-arcs of permanent non-membership*** ($= N_{E_{SC}}$).
    - ***sc-arcs of temporal actual membership*** ($= B_{E_{SC}}$).

The sets of ***sc-elements of an uncertain type***, ***sc-nodes***, ***sc-arcs of a unified type***, and ***sc-files*** are subsets of the set of ***sc-elements***.

The set of ***sc-arcs*** is equal to the union of the sets of ***sc-arcs of permanent membership***, ***sc-arcs of temporal actual membership***, ***sc-arcs of temporal phantom membership***, ***sc-arcs of fuzzy membership***, ***sc-arcs of temporal phantom non-membership***, ***sc-arcs of temporal actual non-membership***, ***sc-arcs of permanent non-membership***.

The sets of ***sc-arcs*** and ***sc-nodes*** do not intersect, that is, they do not have common elements.

The set of ***sc-arcs of permanent membership*** does not intersect neither with the set of ***sc-arcs of temporal actual membership***, nor with the set of ***sc-arcs of temporal phantom membership***, nor with the set of ***sc-arcs of temporal actual non-membership***, not with the set of ***sc-arcs of temporal phantom non-membership***.

The set of ***sc-arcs of permanent non-membership*** does not intersect neither with the set of ***sc-arcs of temporal actual membership***, nor with the set of ***sc-arcs of temporal phantom membership***, nor with the set of ***sc-arcs of temporal actual non-membership***, not with the set of ***sc-arcs of temporal phantom non-membership***.

The set of ***sc-arcs of permanent membership*** does not intersect with the set of ***sc-arcs of permanent non-membership***.

Such types of elements and their correlations are caused by the need to solve the problems of horizontal profile integration by unifying the representation [11], [8], [12], [3].

In order to ensure vertical integration, some elements of the alphabet can be represented by non-atomic information constructions, which can be interpreted as "contents" of these elements [10], [3], [8], [5], for example, based on generalized formal languages. Such elements can be distinguished into a separate type of alphabet elements.

When mapping to generalized formal languages for the purpose of vertical integration (texts), a set of sc-elements (E_{SC}) corresponds to a subset of a generalized formal language with a given alphabet (A_{SC}), just as the set of all pairs of sc-elements corresponds to a subset of a generalized formal language with a given alphabet (A_{SC}).

This Syntactic classification of sc-elements from the ***Syntactic classification of sc-elements of the SC-code Core*** is distinguished by an additional clarification of the syntactic typology of sc-elements.

A. Syntax of the ostis-system internal language

The ***Syntax of the SC-code Core*** corresponds to the syntax of the languages of the unified semantic knowledge representation model (sc-languages) and is set by the ***Alphabet of the SC-code Core*** and two mentioned incident relations of the alphabet elements of these languages by the ***Incidence relation of sc-connectors**** ($= I_{SC}$) and the ***Incidence relation of incoming sc-arcs**** ($= C_{SC}$).

The ***Incidence relation of sc-connectors**** is a binary oriented relation, the first component of each oriented pair of which is some ***sc-connector*** and the second component is one of the ***sc-elements*** connected by the specified ***sc-connector*** with some other ***sc-element***, which is specified in another incidence pair for the same ***sc-connector*** [5], [8].

The set of ***sc-connectors*** is a subset of ***sc-elements*** [5], [8].

The set of ***sc-arcs*** is a subset of ***sc-connectors*** [5].

The ***Incidence relation of incoming sc-arcs**** is a binary oriented relation, the first component of each oriented pair of which is some ***sc-arc*** and the second component is an ***sc-element***, in which the specified ***sc-arc*** is included, i.e. the ***sc-element***, which is the second component connected (linked) by the specified ***sc-arc*** [5].

The ***Incidence relation of incoming sc-arcs**** is a subset of the ***Incidence relation of sc-connectors**** [5].

On the basis of these relations, one incidence relation can be distinguished ***Incidence relation**** $(= R)$. The ***Incidence relation**** is a union of the ***Incidence relation of sc-connectors**** and the backward relation to its symmetric difference with the ***Incidence relation of incoming sc-arcs****.

For each ***sc-connector*** (E), there are two and no more than two pairs of the ***Incident relation of sc-connectors****, the specified ***sc-connector*** is the first binding component. At the same time, for each ***sc-arc*** (A), one of the specified incident pairs must belong to the ***Incidence relation of the incoming sc-arc****.

The ***sc-connectors*** connecting the ***sc-element*** to itself will be called ***loop sc-connectors*** (***loop sc-edges*** and ***loop sc-arcs***). The incidence pairs of ***loop sc-connectors*** are as if they were multiples.

To the ***Incidence relations of sc-connectors**** and ***Incidence relations of incoming sc-arcs**** definition domain, not only ***sc-nodes*** are included but also***sc-connectors***. This means that an ***sc-connector*** can connect (link) not only an ***sc-node*** with an ***sc-node*** but also an ***sc-node*** with an ***sc-connector*** and even an ***sc-connector*** with an ***sc-connector***.

In the sc-text, for each occurrence of an ***sc-element***, its syntactic class can be set (***syntactic class of the occurrence of the sc-element in the sc-text****), according to which syntactic subclasses of occurrences of sc-elements can be distinguished for this sc-text: ***syntactic class of occurrences of terminal sc-elements in the sc-text****(T), ***syntactic class of occurrences of node sc-elements in the sc-text**** (V), ***syntactic class of occurrences of edge sc-elements in the sc-text**** (E), ***syntactic class of occurrences of arc sc-elements in the sc-text**** (A), ***syntactic class of occurrences of basic sc-elements in the sc-text**** (B). For each sc-text, it is possible to determine the set of all its components (***components of sc-text****) and the relation of occurrences of the components of the ***Incidence relation of sc-connectors**** and ***Incidence relations of sc-arcs**** connectives: ***Relation of occurrences of incident sc-connectors of the sc-text**** (I), ***Relation of occurrences of incident sc-arcs of the sc-text**** (C).

In this case, the following features will be performed:

- the sum of powers for unions of sets of ***Relation of occurrences of incident sc-connectors of sc-text**** and ***Relation of occurrences of incident sc-arcs of sc-text**** and sets of ***syntactic class of occurrences of terminal sc-elements in sc-text****, ***syntactic class of occurrences of node sc-elements in sc-text**** and ***syntactic class of occurrences of edge sc-elements in sc-text**** to the number of components of this sc-text (***components of sc-text****);
- the ***syntactic class of occurrences of arc sc-elements in sc-text**** is a subset of ***syntactic class of occurrences of edge sc-elements in sc-text****;
- The ***Relation of occurrences of incident sc-connectors of sc-text**** is a set of pairs connecting the occurrence of a ***component of sc-text**** with the occurrence of a component of the same sc-text, a pair of these components is an element of the ***Incidence relation of sc-connectors****;
- The ***Relation of occurrences of incident sc-arcs of sc-text**** is a set of pairs connecting the occurrence of a ***component of sc-text**** with the occurrence of a component of the same sc-text, a pair of these components is an element of the ***Incidence relation of sc-arcs****;
- The ***Relation of occurrences of incident sc-arcs of sc-text**** is a subset of the ***Relation of occurrences of incident sc-connectors of sc-text****;
- the ***syntactic class of occurrences of basic sc-elements in sc-text**** is a subset of the ***syntactic class of occurrences of arc sc-elements in sc-text****;
- a ***syntactic class of occurrences of arc sc-elements in sc-text**** is a subset of the ***syntactic class of occurrences of edge sc-elements in sc-text****;
- the ***syntactic class of occurrences of arc sc-elements in sc-text**** does not intersect with the ***syntactic class of occurrences of node sc-elements in sc-text****;
- the ***syntactic class of occurrences of arc sc-elements in sc-text**** is a set of pairs connecting sc-text with its component (***components of sc-text****), which is an ***sc-arc***;
- the ***syntactic class of occurrences of node sc-elements in sc-text**** is a set of pairs connecting sc-text with its component (***components of sc-text****), which is an ***sc-node***;
- the ***syntactic class of occurrences of basic sc-elements in sc-text**** is a set of pairs connecting sc-text with its component (***components of sc-text****), which is an ***sc-arc of actual membership***;
- the direct product of the set (***Cartesian product of the set****) of the union of the ***syntactic class of occurrences of terminal sc-elements in the sc-text**** with the difference of the ***syntactic class of occurrences of node sc-elements in the sc-text**** with the ***syntactic class of occurrences of edge sc-elements in the sc-text**** of itself does not intersect with the ***Relation of occurrences of incident sc-connectors of sc-text****;
- for each pair of the ***syntactic class of occurrences of edge sc-elements in the sc-text****, there is at least one pair of the ***Relation of occurrences of incident sc-connectors of sc-text****, the first component of which it is;
- for each pair of the ***syntactic class of occurrences of edge sc-elements in the sc-text****, there are no more than two pairs of the ***Relation of occurrences of incident sc-connectors of sc-text****, the first component of which it is;
- for each pair of the ***syntactic class of occurrences of arc sc-elements in the sc-text****, there is a single pair of the ***Relation of occurrences of incident sc-connectors of sc-***

text*, the first component of which it is;

- for each pair of the ***syntactic class of a of occurrences of basic sc-elements in the sc-text****, there is at least one pair of the ***Relation of occurrences of incident sc-connectors of sc-text****, not belonging to the ***Relation of occurrences of incident sc-arcs of sc-text****, but only if there is another pair of the ***Relation of occurrences of incident sc-connectors of sc-text****, the first component of which it is and the second component of which belongs to the ***syntactic class of occurrence of node sc-elements in the sc-text****.

The ***Syntax of the SC-code Core*** [5] is set:

- by the ***Alphabet of the SC-code Core***;
- by the ***Incidence relation of sc-connectors**** and the ***Incidence relation of incoming sc-arcs****;
- by the rules of connection (incidence) of ***sc-elements*** (for example, which types of sc-elements cannot be incident to each other) when they occur in sc-texts;
- by structural (syntactic) constraints in the semantic neighborhood of the key elements of the ***SC-code Core***.

The syntax of the internal language of ostis-systems (the ***Syntax of the SC-code***) is set by the syntax of the languages for the unified semantic representation of knowledge model (sc-languages) and with the exception of the ***Alphabet of the SC-Code Core*** and the ***Alphabet of the SC-code*** is exactly the same as the ***Syntax of the SC-Code Core***.

The ***Syntax of the SC-code*** [5] is set:

- by the ***Alphabet of the SC-code***, that is, the typology (alphabet) of sc-elements (atomic fragments of SC-code texts);
- by the ***Incidence relation of sc-connectors**** and the ***Incidence relation of incoming sc-arcs****;
- by the rules of connection (incidence) of ***sc-elements*** (for example, sc-elements of which types cannot be incident to each other) when they occur in sc-texts;
- by structural (syntactic) constraints in the semantic neighborhood of the key elements of the ***SC-code***.

B. Basic denotation semantic of the ostis-system internal language

Denotational semantics is a description of the correspondence of information constructions belonging to the language (the ***SC-code Core***) and entities described by these constructions [5].

Denotational semantics of the SC-code Core [5]. According to the unified semantic knowledge representation model, the semantics of the ***SC-code Core*** (including the basic one) is expressed by:

- proper and non-proper sublanguage key elements for (pairs of) languages (language and sublanguage);
- semantic neighborhoods of key elements in sublanguage texts for language pairs;
- semantic interpretations of text elements.

Due to the fact that the semantics and meaning of designations in the texts of languages of the unified semantic representation of knowledge model are expressed through the connections of elements, the family of all texts of a language defines a set of key elements in relation to any of its superlanguages and vice versa. Accordingly, the semantics (interpretations) of text elements and semantic neighborhoods of key elements of the language are designated [8].

Thus, semantics can be defined by enumeration of the key elements of a language (semantic types of elements) or enumeration, formation of its texts. It is important to note that the becoming of language texts in the process of their integration is a natural mechanism that sets not only the denotational semantics of the language, but also the operational and game semantics [9], [23], [24], allowing to consider their joint and unified formalizations within one semantic (meaningful) spaces.

Due to the above and the fact that the integration and processing of knowledge is unthinkable without movement and change, which is a special case of becoming [12], [3], the concept of becoming and the key element of ***becoming**** is a necessary key element of the ***SC-code Core*** and the internal language of ostis-systems (***SC-code***).

The basic mathematical concept that allows expressing the basic denotational semantics of the language in question is the concept of a distensible set (***sc-set***) [8], [3].

Projectively, the concept of an distensible set ($Distensible$) can be expressed in accordance with the scheme [8]:

$$\langle Universe, Events, Becoming, Designator, Distensible, \Box, \mathbf{Z}_{+}, [0;1]\rangle$$

through the mathematical concept of a set as follows:

$$Distensible \tilde{\subseteq} \Xi,$$

where

$$\Xi = 2^{Events} \times \Phi^{Universe},$$

$Universe$ – a set of elements (including designations of distensible sets), $Events$ – a set of (elementary) events, becoming relation $Becoming$:

$$Becoming \subseteq Events \times Events,$$

designation event function $Designator$:

$$Designator \in \left(2^{Events}\right)^{Universe},$$

and

$$\Phi = \bigcup\nolimits_{s}^{s\in\Box} (\Psi)^{\{s\}}; \Psi = \nabla \times \left(\Delta^{Events}\right);$$

$$\nabla = \bigcup\nolimits_{p}^{p\in\mathbf{Z}_{+}} D^{\left(D^{p-1}\right)}; \Delta = \bigcup\nolimits_{q}^{q\in_{+}} D^{\left(D^{q-1}\right)}; D = [0;1],$$

where $\Box$ – a linearly ordered set ($\mathbf{Z}_{+} \subseteq \Box$).

An algebraic structure over distensible sets with operations is permissible:

$$\left\{\bullet_{\Xi}^{k}\right\} \subseteq \Xi^{\Xi\times\Xi},$$

when $k \in \{\cup, \cap, \otimes, \oplus, ...\}$, which can be expressed:

$$\langle\alpha,\delta\rangle \bullet_{\Xi}^{k} \langle\beta,\gamma\rangle = \left\langle \alpha\cup\beta, \tau\left(\left\langle\delta,\gamma,\bullet_{\Phi}^{k}, Universe\right\rangle\right)\right\rangle,$$

where

$$\tau\left(\langle\alpha,\beta,\varphi,\sigma\rangle\right) = \left\{\left\langle\chi,\varphi\left(\left\langle\alpha\left(\chi\right),\beta\left(\chi\right)\right\rangle\right)\right\rangle | \chi\in\sigma\right\},$$

and operations

$$\left\{\bullet_{\Phi}^{k}\right\} \subseteq \Phi^{\Phi\times\Phi}$$

in turn

$$\alpha \bullet_{\Phi}^{k} \beta = \tau\left(\left\langle \alpha, \beta, \bullet_{\Psi}^{k}, \Box \right\rangle\right).$$

The last ones are expresses through operations

$$\left\{\bullet_{\Delta}^{k}\right\} \subseteq \Delta^{\Delta\times\Delta}$$

and mappings:

$$\left\{\bullet_{\nabla}^{k}\right\} \subseteq \left(\nabla^{\left(\Delta^{Events}\right)}\right)^{\nabla\times\nabla}$$

as follows:

$$\langle \alpha, \delta \rangle \bullet_{\Psi}^{k} \langle \beta, \gamma \rangle = \kappa\left(\left\langle \alpha, \beta, \bullet_{\nabla}^{k}, \tau\left(\left\langle \delta, \gamma, \bullet_{\Delta}^{k}, Events \right\rangle\right)\right\rangle\right),$$

where

$$\kappa\left(\langle \alpha, \beta, \varphi, \varepsilon \rangle\right) = \left\langle \varphi\left(\langle \alpha, \beta\left(\varepsilon\right)\rangle\right), \varepsilon \right\rangle.$$

In turn, the remaining operations and mappings are expressed by:

$$\left\{\bullet_{D}^{k}\right\} \subseteq D^{D\times D},$$

$$\zeta \in \left(\bigcup_{p}^{p\in\mathbf{Z}_{+}} D^{\left(D^{p-1}\right)}\right)^{\left(\mathbf{Z}_{+}\times D\right)},$$

$$\nu \in \mathbf{Z}_{+}^{\left(\bigcup_{p}^{p\in\mathbf{Z}_{+}} D^{\left(D^{p-1}\right)}\right)},$$

$$\mu \in D^{\left(\bigcup_{p}^{p\in\mathbf{Z}_{+}} D^{\left(D^{p-1}\right)}\right)}$$

as follows:

$$\alpha \bullet_{\nabla}^{k} \beta = \zeta\left(\left\langle \nu\left(\alpha\right) \cup_{\mathbf{Z}_{+}} \nu\left(\beta\right), \mu\left(\alpha\right) \bullet_{D}^{k} \mu\left(\beta\right) \right\rangle\right),$$

where $\cup_{\mathbf{Z}_{+}} = \max$, and, for example, $\bullet_{D}^{\cup} = \max$, $\bullet_{D}^{\cap} = \min$, $\bullet_{D}^{\otimes} = *$, $\alpha \bullet_{D}^{\oplus} \beta = \left(\alpha \cap^{+} \left(1-\beta\right)\right) \cup^{+} \left(\left(1-\alpha\right) \cap^{+} \beta\right)$.

Moreover:

$$\forall\varphi\forall p\left(\left(\varphi \in D^{\left(D^{p-1}\right)}\right) \to \left(\nu\left(\varphi\right) = p-1\right)\right),$$

$$\forall\sigma\forall p\forall\varepsilon\left(\left(\sigma \in D^{p-1}\right) \to \left(\zeta\left(\langle p, \varepsilon\rangle\right)\left(\sigma\right) = \pi\left(\sigma + \langle\varepsilon\rangle\right)\right)\right),$$

$$\forall\sigma\left(\left(\sigma \in \{1\}^{\nu(\chi)-1}\right) \to \left(\mu\left(\chi\right) = \chi\left(\sigma\right)\right)\right),$$

where:

$$\chi \in D^{\left(D^{\nu(\chi)-1}\right)},$$

$$\pi\left(\langle\rangle\right) = 1,$$
$$\pi\left(s + \langle e\rangle\right) = e \cdot \pi\left(s\right) + \left(1-e\right) \cdot \left(1 - \pi\left(s\right)\right)$$

or in a non-recurrent form:

$$\pi\left(s\right) = \frac{1 + (-1)^{\dim(s)} \cdot \sum_{i}^{i\in\mathbf{N}\cup\{0\}} \frac{(-2)^{i}}{i!} \cdot \sum_{m}^{m\in\left\{s_j | j\right\}^{i}} \prod_{j}^{m_j} m_j}{2}.$$

Structurally, the concept of a distensible set can be expressed within the formal reflexive and descriptive semantics of the languages for the unified semantic representation of knowledge model, with the involvement of more fundamental concepts of the becoming of the actual and phantom (event) [12], [3], [37].

Key elements of the ***SC-Code Core*** [8]:

- an ***sc-sign*** is the designation of the sc-set and the element (component) of any sc-set designated by the sc-element [3], [17];
- ***sc-set***;
  - the sc-set is a distensible set, any sc-element is a designation (sign) of the sc-set;
- ***node sc-set***;
  - a node sc-set is a distensible set that is not an sc-pair of fuzzy membership, any sc-node is a designation (sign) of a node sc-set;
- ***sc-pair of fuzzy membership***;
  - an sc-pair of fuzzy membership is an sc-pair, the designation of which belongs to the relation of fuzzy membership, sc-arc of fuzzy membership is the designation (sign) of the sc-pair, the first component of which is the designation (sc-sign) of the sc-set, denoted by the sc-element from which this sc-arc goes out, and the second component which is the designation (sc-sign) of the sc-set denoted by the sc-element in which this sc-arc comes [12];
- ***sc-pair of permanent membership***;
  - an sc-pair of permanent membership is an sc-pair, the designation of which belongs to the relation of permanent membership, an sc-arc of permanent membership is the designation (sign) of an sc-pair, the first component of which is the designation (sc-sign) of the sc-set S, denoted by the sc-element from which this sc-arc goes out, and the second component of which is permanently (as long as it exists) belonging to the sc-set S designation (sc-sign) of the sc-set designated by the sc-element in which this sc-arc comes [12];
- ***sc-pair of temporal actual membership***;
  - an sc-pair of temporal actual membership is an sc-pair whose designation belongs to the relation of temporal actual membership, an sc-arc of temporal actual membership is the designation (sign) of the sc-pair, the first component of which is the designation (sc-sign) of the sc-set S, denoted by the sc-element from which this sc-arc goes out, and the second component of which is temporarily currently belonging to the sc-set S designation (sc-sign) of the sc-set designated by the sc-element in which this sc-arc comes [12];
- ***sc-pair of temporal phantom membership***;
  - an sc-pair of temporal phantom membership is an sc-pair whose designation belongs to the relation of temporal phantom membership, an sc-arc of temporal phantom membership is the designation (sign) of the sc-pair, the first component of which is the designation (sc-sign) of the sc-set S, denoted by the sc-element from which this sc-arc goes out, and the second component of which is temporarily belonging to the sc-set S designation (sc-sign) of the sc-set designated by the sc-element in which this sc-arc comes [12];

- ***sc-pair of temporal phantom non-membership***;
  - an sc-pair of temporal phantom non-membership is an sc-pair whose designation belongs to the relation of temporal phantom non-membership, an sc-arc of temporal phantom non-membership is the designation (sign) of the sc-pair, the first component of which is the designation (sc-sign) of the sc-set S, denoted by the sc-element from which this sc-arc goes out, and the second component of which is temporarily not belonging to the sc-set S designation (sc-sign) of the sc-set designated by the sc-element in which this sc-arc comes [12];
- ***sc-pair of temporal actual non-membership***;
  - an sc-pair of temporal actual non-membership is an sc-pair whose designation belongs to the relation of temporal actual non-membership, an sc-arc of temporal actual non-membership is the designation (sign) of the sc-pair, the first component of which is the designation (sc-sign) of the sc-set S, denoted by the sc-element from which this sc-arc goes out, and the second component of which is temporarily currently not belonging to the sc-set S designation (sc-sign) of the sc-set designated by the sc-element in which this sc-arc comes [12];
- ***sc-pair of permanent non-membership***;
  - an sc-pair of permanent non-membership is an sc-pair whose designation belongs to the relation of permanent non-membership, an sc-arc of permanent non-membership is the designation (sign) of the sc-pair, the first component of which is the designation (sc-sign) of the sc-set S, denoted by the sc-element from which this sc-arc goes out, and the second component of which is permanently (as long as it exists) not belonging to the sc-set S designation (sc-sign) of the sc-set designated by the sc-element in which this sc-arc comes [12];
- ***node sc-pair***;
  - node sc-pair is an sc-pair designated by the sc-node;
- ***sc-pair***;
  - sc-pair is an sc-set, to which there are only two memberships of different sc-elements or of the same sc-element;
- ***sc-connective***;
  - sc-connective is an sc-set whose sc-subset is an sc-pair;
- ***sc-relation***;
  - sc-relation is an sc-set of sc-connectives;
- ***binary sc-relation***;
  - binary sc-relation is an sc-set of sc-pairs;
- ***slot sc-relation***;
  - slot sc-relation is an sc-set of sc-pairs that are not node sc-pairs;
- ***attributive sc-relation***;
  - attributive sc-relation is an sc-set of sc-pairs of membership (permanent, temporal, actual, or phantom);
- ***sc-file***;
  - sc-file is an entity designated by an sc-element whose contents is an sc-file (a finite dynamic or static data structure);
- ***sc-structure****;
  - sc-structure* is an sc-set in which there is an sc-subset-carrier (the set of primary elements of the sc-structure);
- ***perception ****;
  - perception* is a binary sc-relation between an sc-element and an sc-set of its images;
- ***explanation****;
  - explanation* is a binary sc-relation between an sc-element and an sc-set of its explanations;
- ***becoming****;
  - becoming* is a binary sc-relation between events (states) or phenomena.

Each ***sc-element*** is a sign (designation) of some described entity [5].

Any entity can be designated by an ***sc-element*** and, accordingly, described as a construction of the ***SC-code Core*** [5].

With the help of the sc-elements, it is possible to represent any connections between sc-elements and/or between entities that are designated by these sc-elements. In this case, these connections are considered as extensible sets of connected sc-elements and are designated by sc-arcs, and in the case of non-binary connections – by sc-nodes [5].

Since each sc-connector is semantically interpreted as the designation of a pair of sc-elements connected (linked) by this sc-connector, each pair of incidence of the sc-connector is semantically interpreted as an membership pair connecting the sc-connector with one of the elements of the pair of sc-elements designated by it, and the sc-connector itself is its designation [5]. Any described entity can be designated by an sc-element of an uncertain type, however, the reverse is not true, since some entities can only be designated by sc-arcs of a general type, basic sc-arcs [5].

Basic denotational semantics of the internal language of ostis-systems (SC-code). The basic denotational semantics of the ***SC-code*** (the internal language of ostis-systems) basically corresponds to the basic denotational semantics of the ***SC-Code Core***, due to the preservation of all the key elements of the ***SC-Code Core*** in the SC-code. However, the semantics of the sc-texts of the ***SC-code*** differs from the semantics of the sc-texts of the ***SC-code Core***, since a more precise semantic interpretation can be set by the membership of the sc-text component to the corresponding syntactic class that does not belong to this sc-text, in contrast to its assignment by belonging to the key element, belonging to sc-text of which is required [5], [8].

The semantic proximity of the ***SC-code*** and the ***SC-code Core*** is a consequence of the fact that the ***SC-code*** is a syntactic extension of the ***SC-code Core*** [5].

V. Semantic Space

A. Review of approaches

The word "space" takes its etymological roots in Proto-Indo-European word with the meaning "to stretch" or "to pull". The word "semantics" originates from Anc. Greek "semantikos" that is created by union of "semaino" (to indicate, to sign) and suffix "ikos". On the other hand, "meaning" rooted in "semaino" originates from the Proto-Indo-European word with the meaning "to think" or "to change". So, "semantic space" means "stretched thought" and is interpreted as a phenomenon of thinking (i.e. movement of thoughts). The need for considering such a concept is related to the need for analysis of structural and quantitative (metrical) features aiming to detect limits, assess completeness of thinking processes, and to optimize costs needed per their modeling having finite resources.

It probably might seem, that the concept of "semantic space", or something close or similar to this, originates from ancient times by philosophers, including idealists and dualists, e.g. Platonic "world of ideas" [38]. However, philosophic concepts change along their development and are often not well defined so that one can make unambiguous statements and be definitely certain about them, particularly being in the conditions of incomplete information about historical "facts" and inaccuracy of historical evidence. In addition, the concept of "space" often related to the concept of "time", for it most likely is a "container" ("substance") for something changeable, impermanent, i.e. – materialistic, things (from "thinking") rather than constant, "perpetual" "ideas". Perhaps, R. Descartes was one of the first in European history who showed signs of intentions, which came down to us, to connect "thinking" and "material" ("things") "space" via the God and relationism [39] that was stood by G. W. Leibniz afterwards [40], and together with the similarity of properties of "thinking" to natural "extension" is reflected by D. Hilbert [41]. Reasoning about "space" and "thinking", D. Hilbert had been rejecting their infinity. Also, it could be that Hegel's concepts [37] are closer to the concept of "semantic space". In dialectical materialism in accordance with the definition for "matter" the existence of meaning should be accepted only materialistically in the space and time.

Largely due to the development and definition of mathematical concepts for various mathematical structures, in modern science, "space" is understood not only as "material", "physical space".

Let us take a look at the history of the usage of the term "space" in mathematics. It is believed that infinite constructions are not considered in the beginnings of Euclid, which is also considered characteristic of ancient mathematicians, therefore there are no sufficient grounds to speak of an "ancient concept" of"Euclidean space". Modern science II defines a lot of mathematical structures (Fig. 10) up to hypothetical ones ("antispace" [47]) in the name of which the term "space" is used:

- vector (linear) space (/finite-dimensional) [48];
- affine space (/finite-dimensional);
- topological space [49];
- linear topological space;
- pseudometric space [50];
- metric space [51];
- locally convex space;
- semi-normed space;
- normed space [54];
- Banach space;
- pre-Hilbert space;
- Hilbert space;
- function space;
- Euclidean space;
- pseudo-Euclidean space;
- space with measure;
- probability space.

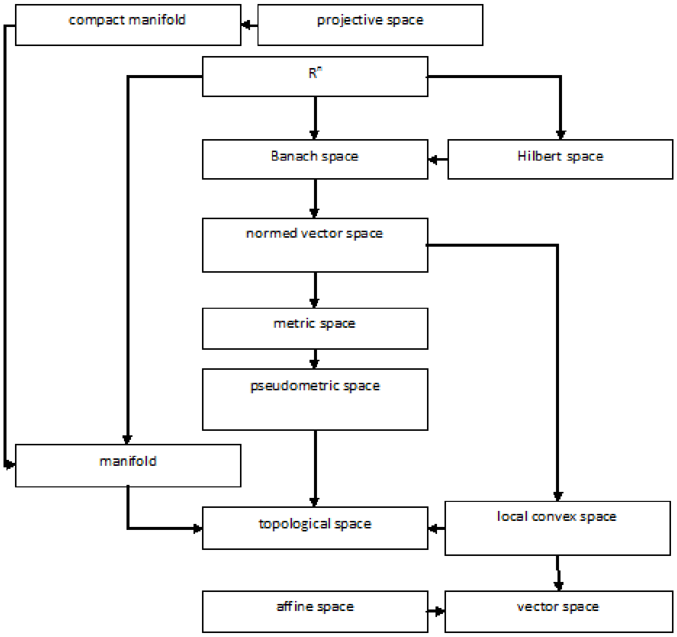


Fig. 10. A conceptual scheme of "spaces".

In other sciences, it is possible to find "phase space" [52], "object space" [53], etc. Among the representatives of more exact sciences, the works of D. Bohm and V. V. Nalimov can be highlighted.

The works of D. Bohm [43], [42] raise questions of interpretation of physical phenomena at the level of the microworld within the subject of quantum mechanics, questions related to the mutual influence of an observer (subject) on an object in an experiment, questions related to integrity and partiality of perception, the physical nature of consciousness and the duality of the properties of the objects of the microcosm, revealed in the experiments II. For D. Bohm, "holonomic movement" and "implicate order" are conceived as a principle and substance capable of linking "part" and "whole", "consciousness" associated with "field of knowledge" (thougths), considered as a process, and "matter" in space, which has higher dimensions. The most complete work in this regard is his work "Wholeness and the Implicate Order" [43].

In the works of V. V. Nalimov [44], [45], the "probabilistic space of meanings" is considered within the "probabilistic theory of meanings". According to the position of one of the founders of the theory of probability, A. N. Kolmogorov, one of its fundamental questions is the question of the connection and mutual influence of the subject on the object [71]. Despite the fact that the concept of a probability space could hardly be considered as a model for the world of ideas in Plato's time, V. V. Nalimov considers himself a Platonist. Such a "probability space of meanings" is assumed to be different from the "physical space" and not included in it II:

"Physical space, according to Bastin's ideas, represents a finite series of points, for which the rules for constructing new points are postulated, creating a hierarchy of points (see [215], p. 99)." [46]

Table II
Comparison of approaches to investigate «semantic spaces»

	exterior (physical) approach	*interior (abstract, logic-semiotic) approach based on analysis of*	
		quantitative features (probabilistic (additive) measures)	*structure-dynamic features*
cognitive process analysis (introspection)	+	-	?
adaptation	+	-	+
unification	-	+	+

In modern works in the technical sciences [14], perhaps the closest concepts are those expressing the meaning of the term "semantic space" (interior approach II). Common in many approaches to working with "semantic space" is the consideration of word forms or lexemes (sets of word forms) and their features (II). The following approaches III are found in the literature [14]:

- an approach based on semic axes and feature space (binary $\{0,1\}^n$, unipolar $[0;1]^n$, bipolar (bisemic) $[-1;1]^n$);
- an approach based on semic axes and neural encoding of a place field recognition of meanings (while words and phrases have areas (subsets) of meanings being connected by other parts of speech as inclusion and intersection, texts correspond to the path of connected areas, and binary coding is used for groups of neurons recognizing meanings);
- an approach based on the "sense-text" [55] model (reflection of the incompleteness of semantic scales and analysis of syntagmas and surface-syntactic structure);
- an approach based on neurolinguistic data that reflects the processes of production and perception of speech in neural networks (lexical production network), is close to the "meaning-text" model;
- models built on the basis of static analysis (of corpora) of texts (vector space model).

The statistical approach to natural language processing is opposed to the intuition and communicative experience of scientists [14].

An approach is based on the semantic statistical hypothesis that the meaning of words (lexemes) is determined by the context of usage (its statistical pattern) in the language (with a communicative structure) [14].

Vector space models of semantics [14]. A concrete model is considered for two cases: a large vocabulary case ($N \leq M$) and an information retrieval case ($M \leq N$), where M is the dictionary size, N is the number of contexts.

On the basis of statistics, a matrix of dimensions $M \times N$ of frequencies p_{ij} of occurrences of a lexeme (word) w_i in document (context, subtexts that may overlap) c_j.

$$x_{ij} = \max\left(\{0\} \cup \left\{\log\left(\frac{p_{ij}}{(\sum_j p_{ij})*(\sum_i p_{ij})}\right)\right\}\right)$$

The denominator contains word and context probability estimates, respectively.

In the case of a nondegenerate matrix with its rank $r = N$, each such matrix defines a point in the Grassmannian of N-dimensional subspaces of a M-dimensional space ($N \leq M$).

In the case of a nondegenerate matrix with its rank $r = M$, each such matrix defines a point in the Grassmannian of M-dimensional subspaces of a N-dimensional space ($M \leq N$).

Each text is a point in the Grassmannian [56] corresponding to the projective space $P^{M-1} = Gr\left(\langle 1, M\rangle\right)$, relative to one selected context. For all contexts, in accordance with the order of contexts in texts and the resulting oriented N-tuple, a route (path) can be constructed by connecting adjacent points in the N-tuple with geodesics. For two texts T and T', these paths will be two polygonal curves. The Frechet metric between these curves [18] can be calculated using the Fubini-Study metric [19] in P^{M-1}. To calculate it, the paths $\Gamma\left(\mathrm{T}\right)$ and $\Gamma\left(\mathrm{T}'\right)$ should be parameterized through t ($\gamma \in \Gamma(\mathrm{T})^{[0;1]}$, $\gamma' \in \Gamma(\mathrm{T}')^{[0;1]}$):

$$\delta\left(\langle\Gamma\left(\mathrm{T}\right), \Gamma\left(\mathrm{T}'\right)\rangle\right) = \inf_{\gamma,\gamma'} \max_{t \in [0;1]} \left\{d_{FS}\left(\langle\gamma\left(t\right), \gamma'\left(t\right)\rangle\right)\right\}.$$

Another way to specify a linear order is to consider the flag (filtering (flag manifold)) [57] in $\mathbf{R}^M$ defined by expanding (embedding) contexts. As a result, for the text we get points (flags) in the flag manifold. For flag varieties, one can also calculate the Fubini-Study metric [19].

This order corresponds to the temporal dimension (communication process in time), which can be significant. Other ordering may be independent of this, such as alphabetical or Zipf's order [58], [59].

The issues of formalizing meanings, their correlation, the genesis (in space and time) of languages are considered in the works of V.V. Martynov [60], [61], [62].

To solve the identified problems, it may be worth turning to an alternative approach: to explore not only the communicative

Table III
Comparison of “semantic spaces” construction approaches

	semic axes and		*“meaning-text” model*	*neurolingvistic coding*	*statistical model (semantic vector space model)*
	feature spaces	*neural encoding place field recognition*			
defined semic axes	+	+	-	-	-
dynamic (computing) decomposition	-	+	-	+	-
cognitive process analysis (introspection)	-	+	-	+	-
accounting of NON-factors (incompleteness)	-	-	+	+	+

structures of the language but also the cognitive-representational structures of the language [70].

The prerequisites for building knowledge representation models that claim to be universal have been created in the works carried out in accordance with the graph-dynamic paradigm of information representation and processing [17], [5], [8] .

B. SC-space

The concepts of ***SC-space*** and ***SC-code*** are necessary to clarify and formalize the meaning of information structures with the unification of the semantic representation of information. The meaning of an information construction is ultimately determined by (1) the internal connections of all elementary fragments of this construction and (2) its external connections with the elements of the semantic space (its position in the context). The semantic space is the result of the natural formation of knowledge in the process of their integration.

The most important advantage of ***SC-space*** is the possibility of clarifying such concepts as the concept of similarity (similarities and differences) of various described “external” entities, similarity of unified semantic networks (texts of the ***SC-code***), the concept of semantic proximity of the described entities (including texts of the ***SC-code***).

It should be noted that it cannot be ruled out that the union of two arbitrary texts of such languages will not be the text of the language of the unified semantic knowledge representation model due to the abstractness of the languages of the unified semantic knowledge representation model and the conventionality of the choice of labels for the elements of their texts. To avoid the results of such eclectic combinations in terms of syntax or semantics, a set of “semantic spaces” should be considered for abstract languages. However, it may be sufficient to consider one “semantic space” for specific (real) languages.

Next, consider:

- possibility of transition from sc-texts to graph structures and from them to topological space;
- the ability to move from sc-texts to graph structures and from them to a manifold (topological space);
- possibility of transition from sc-texts to graph structures and from them to metric space.

On the set of elements that form ***SC-space***, it is possible to study topological properties and consider ***SC-space*** as a topological space. It should be noted that despite the fact that the ***SC-code*** is focused on the semantic representation of knowledge, due to the presence of non-factors, not all meanings can be represented at some point in time while the structure of the corresponding representation is unknown. Therefore, the structural and topological properties of the texts of the knowledge representation language rather determine the syntactic space than the semantic (meaning) one. Although both can approach each other as the uncertainties caused by non-factors are eliminated.

It is necessary to make several transitions to get the topological space as a transformation of the sc-text, which could result from the integration of many smaller sc-texts:

- transition from texts with syntax of sc-languages to a pseudograph (oriented or unoriented) (Fig. 6, Fig. 7, and Fig. 8);
- transition from an oriented pseudograph to a transitive oriented pseudograph;
- transition from an oriented (transitive) pseudograph to an oriented (transitive) graph;
- transition from an oriented transitive pseudograph to a topological space;
- transition from an oriented pseudograph to an unoriented graph;
- transition from an unoriented graph to a manifold (topological space).

There is a transition from an oriented pseudograph to an oriented bipartite graph (Fig. 11). During this transition, the following occurs: mapping vertices to edges and arcs; mapping arcs to connections (connectives) in accordance with the direction of arc orientation.

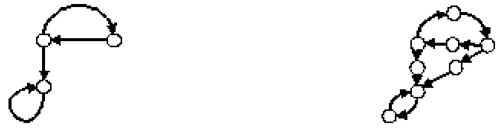


Fig. 11. Transformation of an oriented pseudograph to a bipartite orgraph.

There is a transition from an oriented pseudograph to a transitive oriented pseudograph. (Fig. 12).

In this transition, transitive closure of arcs is performed.

There is a transition from a (transitive) oriented pseudograph to a (transitive) oriented graph (Fig. 13).

With this transition, loops are eliminated.

There is a transition from an oriented pseudograph to an unoriented graph (Fig. 14).

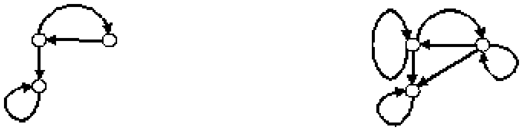


Fig. 12. Transformation of an oriented pseudograph to a transitive oriented pseudograph.

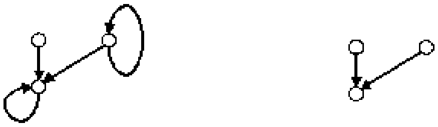


Fig. 13. Transformation of a (transitive) oriented pseudograph to a (transitive) orgraph.

In this transition, the following is carried out: matching vertices to vertices; matching pairs of triples of vertices and triples of edges to arcs.

Fig. 14. Transformation of an oriented pseudograph to an unoriented graph.

There is a transition from an unoriented graph to a manifold (topological space) (Fig. 15).

The following is carried out in this transition: matching figures (points) to vertices; matching figures (lines, two-point sets) to edges.

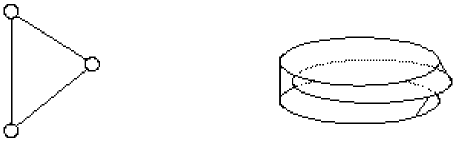


Fig. 15. Transformation of an unoriented pseudograph to a manifold.

The transition from a transitive oriented graph $G = \langle V, E\rangle$ to a topological space T is [63]:

$$G \to G\downarrow \to \overline{T} \to T;$$

$$G\downarrow = \langle V, E\downarrow\rangle;$$

$$E\downarrow = \{g\downarrow | g \in E\};$$

$$E\downarrow = \left\{E^{-1}(v) \cup \{v\} | v \in V\right\}.$$

Let us consider the concept of specialization for the reverse transition as a transitive relation (x is the specialization of y):

$$x \in \overline{\{y\}}^{T},$$

that is, x belongs to the closure of $\{y\}$ in T.

There is a reverse transition:

$$E = \left\{\langle x, y\rangle \middle| x \in \left(\overline{\{y\}}^{T} / \{x\}\right)\right\}.$$

Let us study the question of the possibility of considering ***SC-space*** as a metric space.

The syntactic metric is specified on the lines of the generalized formal language in accordance with the metric tensor over the identified matches, transpositions, exhanges, duplications, fusions, generations, and deletions [13], [3]. The distance ρ_p between the generalized strings α and β is:

$$\rho_p(\langle\alpha, \beta\rangle) = \rho_p^0(\langle\alpha, \beta\rangle); (16)$$

$$\rho_p^k(\langle\alpha, \beta\rangle) = \begin{cases} 1 | \langle\alpha, \beta\rangle \in A \times A \\ \psi_k^\beta\left(\rho_p^k(\langle\alpha, \langle\beta\rangle\rangle)\right) \left|\langle\alpha, \beta\rangle \in \left(A^{(*^*)}/A\right) \times A\right. \\ \psi_k^\alpha\left(\rho_p^k(\langle\langle\alpha\rangle, \beta\rangle)\right) \left|\langle\alpha, \beta\rangle \in A \times \left(A^{(*^*)}/A\right)\right. \\ \varphi_p^k(\langle\alpha, \beta\rangle) \left|\langle\alpha, \beta\rangle \in \left(A^{(*^*)}/A\right) \times \left(A^{(*^*)}/A\right)\right. \end{cases}$$

$$\psi_k^\chi(\gamma) = \upsilon_k^\chi * \gamma; (17)$$

$$\varphi_p^k(\langle\alpha, \beta\rangle) = \sqrt[p]{\sum_{i=1}^{\dim(\alpha)} \sum_{j=1}^{\dim(\beta)} \psi\left(\left\langle \varepsilon_\beta^\alpha(i), \left(\rho_p^{k+1}(\langle\alpha_i, \beta_j\rangle)\right)^p, \omega_{ijk}^{\varepsilon_\beta^\alpha(i)}\right\rangle\right)};$$

$$\psi(\langle\delta, \lambda, \chi\rangle) = \chi * \left(1_{\{M,R,X,T,P\}}^{\{\delta\}} * \lambda + 1_{\{I,D,G,C,E,F\}}^{\{\delta\}}\right);$$
(18)

$$1_\lambda^\gamma = \begin{cases} 0 | \emptyset = (\lambda \cap \gamma) \\ 1 | \emptyset \subset (\lambda \cap \gamma) \end{cases}, (19)$$

$$\varepsilon_\beta^\alpha \in \{M, T, R, I, D\}^{\{i | (i \in (\mathbf{N}/\{0\})) \wedge (i \leq \max(\{\dim(\alpha)\} \cup \{\dim(\beta)\}))\}},$$

where υ_k^χ, $\omega_{ijk}^{\varepsilon(\langle\alpha,\beta\rangle)(i)}$ are weight coefficients, p is a parameter.

Therefore, the syntactic metric and the metric space are defined in a natural way in the case of representation of sc-texts by texts of a generalized formal language.

It is necessary to take into account the semantics of sc-elements and structures from them in all its forms, i.e. denotational, operational, and others [9], [23], [24], for the purpose of constructing a "semantic metric" ("semantic space metric"). Thus, it is also necessary to consider not only structures and their elements (sc-elements) but also the becoming [12] of structures and their elements in the processes of accumulation and integration of knowledge as well as models in which their specification is possible [15], [16]. To do this, we turn to models of knowledge specification and knowledge integration as well as models that can generalize these models for calculating semantic metrics for limit type structures.

C. Knowledge specification model

The knowledge specification model [8] is given by a set of (finite) formal models of (finite) fragments ontologies [4] of knowledge bases (KB) Z:

$$Z = \{\langle G, R, O\rangle\},$$

G is a finite non-empty set of designations in a KB fragment, R is an oriented finite set of relations on designations in a KB fragment, O is an oriented finite set of designations interpretation functions in a KB fragment.

$$\left\langle Z \cup 2^{\left\{\omega(z) | z \in Z^2\right\}}, 2^{\Omega} \right\rangle,$$

where 2^{Ω} is a set of relations of the specification model and ω is a function of ontological model elements:

$$\Omega = \bigcup\nolimits_{\{x\} \cup \{y\} \subseteq Z} \left(\{x\} \times \{y\}\right) \times 2^{\omega(x) \times \omega(y)};$$

$$\omega\left(z_i\right) = G_i \cup \left\{r \left| r = R_{ij} \right.\right\} \cup \left\{o \left| o = O_{ij} \right.\right\} \cup \left\{k \left| k \in R_{ij} \right.\right\} \\ \cup \left\{p \left| p \in O_{ij} \right.\right\} \cup \left\{a \left| \langle a, v \rangle \in O_{ij} \right.\right\}.$$

The knowledge specification model considers the semantics of knowledge for pairs of knowledge base fragments on finite structures within knowledge specification relations, the power of which can be unlimited. As a result, all problems of semantics analysis within the knowledge specification model are solvable for any pair of knowledge base fragments.

D. Knowledge integration

The [8] knowledge integration model is aimed at solving problems of continuous horizontal-profile knowledge integration and concludes a set of (finite) knowledge base fragments $J \subseteq 2^{V \cup E}$, where V is a set of designations (sc-elements), E is a set of connectives of designations incidence relations V.

The integration model concludes four types of relations:

- relations of ontological comparison, which for a pair of KB fragments allow obtaining a set of relations (sc-relations, comparisons) of similarity and difference, which have the property of reflexivity or irreflexivity $R_M \subseteq (J \times J) \times \Xi$;
- the fusion relation $R_F \subseteq (J \times J) \times 2^{V \times V}$;
- the (designations) mapping relation (embedding, inclusion) $R_O \subseteq J \times J$;
- the integration relation $R_I \subseteq (J \times J) \times J$;

These four types of relations define the order of solving knowledge integration problems:

- first, the similarity and difference of designations in the original fragments (texts) should be determined by alignment and comparison (in accordance with the knowledge specification model);
- then, the pairs of matching designations must be localized and fused;
- then, the mapping must be found as a mapping of each original fragment to a fragment containing the designations resulting from the merger and the remaining designations of the original fragments;
- then a fragment should be formed, which is the result of integrating a pair of original fragments, that is, a fragment onto which each of the original fragments is mapped.

The process of solving this problem is expressed in the becoming of designations as a result of merging and in the becoming (formation) of integrated fragments (texts) resulting from the integration of the original fragments.

If we trace the branches of the relation of becoming of integrated fragments (texts) from the original fragments along the mapping relation (embedding), then we can see that the formation of integrated texts generates the movement in the direction of knowledge accumulation as a natural "arrow" that regulates memorization processes. The later allows defining an ordinal scale on such texts, which sometimes may be internal: «arrow of time».

The formation (becoming) of integrated texts preserves the connectives of incidence in their structure, ensuring the convergence of structures to some integrated substructures (of a "space"), which allows proposing and establishing a (semantic) metric for such substructures which are the limit type substructures.

E. Semantic space metamodel

In accordance with the model of events and phenomena [12] and the relation of becoming [3], [37], let us consider linearly ordered sequential unions of non-intersecting chains of the generalized relation of the designations mapping in the unions of texts of languages of the unified semantic knowledge representation model.

To each sequential union, let us associate the numbering function on the universal linear scale $\Box$ of the elements F ($F \subseteq J$) connected by the edges of the chain, in the order of the edges of this chain.

Let us single out a subclass of historically finite linearly ordered unions of disjoint chains of the generalized designations mapping relation.

Let us single out a subclass of locally finite linearly ordered unions of disjoint chains of the generalized designations mapping relation.

A subclass of finite linearly ordered unions of disjoint chains of a generalized relation is the intersection of the subclasses of historically finite linearly ordered unions of disjoint chains of the generalized designations mapping relation and locally finite linearly ordered unions of disjoint chains of the generalized designations mapping relation.

For the subclasses, we also consider the same numbering function as for the entire class of sequential unions. The corresponding functions will be called «histories»: $H = F^{\Box}$. Accordingly, let us single out the historically finite histories $R_{HISTORICFINITE} \in S_H$, the local finite histories $R_{LOCALFINITE} \in S_H$ and the finite histories $R_{FINITE} \in S_H$; $R_{FINITE} = R_{HISTORICFINITE} \cap R_{LOCALFINITE}$. Let us also single out:

- the subclass of well-ordered histories to the beginning $R_{TOTALBACKWARD} \in S_H$;
- the subclass of well-ordered histories to the end $R_{TOTALFORWARD} \in S_H$;
- the subclass of well-ordered histories (two-sided) $R_{TOTAL} \in S_H$; $R_{TOTAL} = R_{TOTALFORWARD} \cap R_{TOTALBACKFORWARD}$.

The metamodel is given by the pair:

$$\langle H, S_H \rangle,$$

where $S_H \subseteq 2^{(H^*)}$. Let us single out the following relations of the semantic space metamodel:

- the subhistory relation $R_{SUB} \in S_H$;
- the superhistory relation $R_{SUPER} \in S_H$ (inverse to the subhistory relation $R_{SUPER} = (R_{SUB})^{-1}$);
- the continuous subhistory relation $R_{SUBCONTINOUS} \in S_H$; $R_{SUBCONTINUOUS} \subseteq R_{SUB}$;
- the continuous superhistory relation $R_{SUPERCONTINUOUS} \in S_H$ (inverse to the continuous subhistory relation $R_{SUPERCONTIONUOUS} = (R_{SUBCONTIONUOUS})^{-1}$);
- the relation of the initial subhistory $R_{START} \in S_H$;
- the relation of the final subhistory $R_{FINAL} \in S_H$;
- the history convergence relation $R_{CONVERGENCE} \in S_H$ (two sequences of becoming (of) integrated texts converge if they have a common final history $R_{CONVERGENCE} = \Big((R_{FINAL})^{-1} \circ R_{FINAL} \Big) \cup \Big(R_{FINAL} \circ (R_{FINAL})^{-1} \Big)$);
- the relation of the maximum well-ordered subhistory to the beginning $R_{FORWARD} \in S_H$; $R_{FORWARD} \subseteq R_{START}$;
- the relation of the maximum well-ordered subhistory to the end $R_{BACKWARD} \in S_H$; $R_{BACKWARD} \subseteq R_{FINAL}$;
- the relation of maximal linearly ordered strict subhistory $R_{EDGE} \in S_H$ (the edge relation $R_{EDGE} \subset R_{SUB}$);
- the relation of minimal linearly ordered strict superhistory $R_{FACE} \in S_H$ (the face relation $R_{FACE} \subset R_{SUPER}$, inverse to the edge relation $R_{FACE} = (R_{EDGE})^{-1}$);
- the enclosure relation $R_{ENCLOSE} \in S_H$;
- the disclosure relation $R_{DISCLOSE} \in S_H$, inverse to the enclosure relation $R_{DISCLOSE} = (R_{ENCLOSE})^{-1}$;
- the relation of possibility of interaction (interoperability) $R_{INTEROPERABILITY} = R_{DISCLOSE} \circ R_{ENCLOSE}$.

The following relations are reflexive: $R_{ENCLOSE}$, $R_{DISCLOSE}$, R_{START}, R_{FINAL}, R_{SUB}, R_{SUPER}, $R_{SUBCONTINUOUS}$, $R_{SUPERCONTINUOUS}$.

For locally finite histories h, the following is true:

$$\forall \iota \left((\iota \in \Box) \to \left(|h(\iota + 1) / h(\iota)| \in \mathbf{N} \cup \{0\} \right) \right).$$

Sequences of integrated texts (histories) can be embedded in some (unified) "semantic space".

Such substructures as subspaces can be distinguished within spaces of various kinds (topological, vector, metric).

The metric of the metric semantic space can be constructed for the structures of the metamodel of the semantic space. For ontologies in which all NON-factors of extensional knowledge (the closed world assumption) can be eliminated during some finite history, it is possible to construct a space with a metric by introducing for each designation of a set (sc-set) a characteristic vector (or matrix (i.e. vector of vectors or matrices)). Such a vector characterizes inclusion in the history of the designation of the set (sc-set) or the possible occurrence of the designation of this set (sc-set) and contains all the corresponding components for each designation of the set (sc-set) or the possible occurrence of its element or attribute of such an occurrence. The order of the components of the vector is coordinated with the order of the becoming of designations in history. Events, values of fuzzy measures are considered as attributes of occurrences. When calculating the semantic metrics (metrics of the semantic space) for vectors, the vector of modules of the difference of their components is calculated. Next, a certain norm of the vector is calculated, which allows introducing a semantic metric. For such ontologies with a (finite) set of attributes, the metric can be calculated similarly due to the one-to-one correspondence:

$$\left(A^B\right)^C = A^{B \times C}.$$

The above is also true if the components of the characteristic vector take values on the interval $[0; 1]$, that is, if there is such a non-factor as fuzziness (according to L. Zade). When attribute values do not belong to this interval, it is required to reduce attributes (sc-sets) to attributes (sc-sets) in canonical form, whose values belong to this interval.

In the case of the presence of uncertainty (the open world assumption) as a non-factor of knowledge, one can turn to the apparatus of rough sets [6]. Due to the presence of necessary operations in the algebra of distensible sets (sc-sets), for a pair of sets and for each component of the characteristic vector, it is possible to define an upper and lower estimates (by analogy with rough sets [6], [7]), which correspond to a component of some distensible subset (of an distensible superset). Further, according to the approaches for rough sets, it is possible to calculate the maximum (upper) and minimum (lower) vectors of the modules of the differences and calculate their norms. It should be noted that if one can obtain a metric space based on the norm of the upper vector then, in the worst case, one can obtain only a family of pseudometrics based on the lower vector and its norm [50]. However, one can find some pseudometric not exceeding of the value of none of this family for any pair of elements. Thus, in this case, one (or more) pairs of spaces can be obtained on one carrier, the first of which will be metric (and pseudometric) (the metric estimate is from above), and the second is pseudometric (the metric estimate is from below). With further accumulation of knowledge, as the mentioned non-factor (uncertainty) is eliminated, both spaces may converge to each other in terms of the value of pseudometrics until the values of the pseudometric (lower estimate) coincide with the values of the metric (upper estimate).

Since the transition from finite structures to non-finite structures consists of a sequence of steps, it takes time (potential infinity). This formation, in turn, is connected with the formation of integrated texts, which, in turn, is associated with their operational semantics. Thus, the very process of formation (becoming) of integrated texts (integration) determines the metrics of the semantic space.

Is the metric for designations with potentially unbounded (reflexive) semantics computable?

In the case when there is confidence that the histories are not represented either as historically finite histories or as locally finite histories, but the rules or the mechanism for generating histories to calculate (possibly only with the engagement of hypercomputing) the metrics are known, the following grammatical rules can be used.

$$0 \to XXX$$
$$1 \to YYY$$

$$YYYXXX \to YXYXYX$$
$$XXXYYY \to YXYXYX$$

$$YXYXYXYXYXYX \to YXYXYX$$
$$YYYYXYXYXXXX \to YXYXYX$$
$$XXXYXYXYXYYY \to YXYXYX$$

$$YYYYXYXYXYYY \to YYYXXXYYY$$
$$XXXYXYXYXXXX \to XXXYYYXXX$$

$$YXYXYX \to \varepsilon$$

$$XXX \to \varepsilon$$
$$YYY \to \varepsilon$$

. They can be used as rules of game [23], [24]. It is possible to calculate some value of the metric by counting the number and remembering the order in which these rules are applied to obtain a given set of vectors in accordance with the semantics of such a game.

In a finite time, it is practically impossible without hyper(super)computations to distinguish (semantically) a knowledge base with finite semantics from a knowledge base with potentially unlimited semantics.

Designations with potentially unlimited semantics can be represented as structures with finite semantics, for which the metric is defined with the accuracy of a given interval.

F. Space-time and semantic space order

The objects and connections of the subject domain, as well as the texts themselves, are assumed to be located in physical time and space (space-time [64], [65], [14]). It is considered inappropriate if the complexity of the spatio-temporal structures presented in the texts significantly exceed the complexity of the structure of physical space and time. Elementary events are connected by the relation of becoming. These events are assumed to correspond to the elements (points) of space-time. Thus, they are represented and explicitly expressed in the semantics of designations in the texts of the languages of the model of the unified semantic representation of knowledge.

This curve can be a curve of the second order (a quadric). Any graph can be represented by geometric shapes in three-dimensional space without intersections of shapes that would correspond to its non-incident vertex or edge. For example, it is in the case of (1) representation of graph vertices in three-dimensional space by straight lines intersecting a parametrically given convex curve lying in a plane along a given (not forming a cycle) step of the parameter and also perpendicular to this plane at intersection points and (2) representation of arcs by straight lines intersecting pairs of straight lines representing vertices. Each line lies in its own plane parallel to the plane in which the convex curve lies. If there is a metric space for the physical space-time then it is possible to define the metric of elementary events with the metric space on them for the knowledge base.

Distensible sets can be considered as the average of the set of points of elementary events, then the space-time metric for the pair of designations for distensible sets s and n:

$$\mathrm{M}\left(\langle s, n\rangle\right) = \left(\frac{\sum_{x=1}^{|V(s)|} \sum_{y=1}^{|V(n)|} d\left(\left\langle V(s)_x, V(n)_y \right\rangle\right)}{\left|V\left(s\right)\right| * \left|V\left(n\right)\right|}\right),$$

where V is a function of the feature vector (points of elementary events), d is the metric of points of elementary space-time events.

In general, the relation of becoming is not antisymmetric, but its condensation can set the order, which can be represented by structures formed in the processes of semantic logging [3], [66]. Recording the processes [66] of integration (fragments) of texts sets the internal (temporal) order and allows you to reverse these processes along each of the branches of the formation of integrated (fragments) of texts. It is assumed that this order corresponds to the temporal order in space-time.

VI. Application for problem solving of taxonomy optimization

Let’s consider the application of concepts related to the concept of a metric space to solve the problem of optimizing the representation of taxonomies This task lies in the fact that it is necessary to minimize the number of operations on the states of the taxonomy knowledge processing model and the number of permanently stored classes. If we consider taxonomy classes as classical sets, then for their expression we can choose the operational basis of set-theoretic operations. They are the operation of removing the taxonomy class and constructive operations of the algebra of sets: intersection and symmetric difference ($\cap$, $-$). Examples of other operational bases of set algebra are set difference with union or intersection of sets. In the mentioned basis, the union of sets is expressed:Examples of other operational bases of set algebra are set difference with union or intersection of sets. The union of sets is expressed through the mentioned basis as follows:

$$A \cup B = (A \cap B) - A - B.$$

In addition to the operational basis, a model basis is considered. The model basis is the minimum number of taxonomy classes through which any of its classes can be expressed. Thus, the taxonomy is defined in accordance with the structural approach by two sets: an initial state, a set of classes and a set of rules (operations), with which it is possible to get any family of taxonomy classes Let’s define the concept of closure of a taxonomic structural model (operational-model closure) in order to formulate the problem: $\overline{\langle\{\sigma\}, \lambda\rangle} = \left\langle\left(\bigcup_{\varphi\in\lambda}\varphi\right)^{\circ}(\sigma), \left(\bigcup_{\varphi\in\lambda}\varphi\right)^{\circ}\right\rangle$, where $\sigma \subseteq \theta$ is a subset of the set of all taxonomy classes θ, $\lambda \subseteq \left(2^{\theta}\right)^{\left(2^{\theta}\right)}$ is a set of operations on taxonomy classes.

Let's consider the set of all possible transitions $\bigcup_{\varphi\in\lambda}\varphi$ on the set of operations λ.

Task.

Given:

$$\Gamma \subseteq 2^{\bigcup_{\varphi\in\lambda}\varphi};$$

function of useful output from job (information capacity):

$$\rho \in \delta^{2^{\theta}\times 2^{\theta}};$$

cost function of (time) resources for job:

$$\tau \in \delta^{2^{\theta}\times 2^{\theta}};$$

functions:

$$\pi \in \delta^{\{|\gamma||\gamma\in\Gamma\}};$$

$$\psi \in \delta^{\{|\gamma||\gamma\in\Gamma\}};$$

α, β.

Required:

$$\alpha * |\sigma| + \beta * |\lambda| \to \min;$$

$$\overline{\langle\sigma,\lambda\rangle} \to \max;$$

$|\Gamma| \to \max$ for any $\gamma \in \Gamma$ to satisfy

$$\sum\nolimits_{\chi\in\gamma}\tau\left(\chi\right) \le \pi\left(|\gamma|\right) * \sum\nolimits_{\chi\in\gamma}\rho\left(\chi\right);$$

$$\sum\nolimits_{\chi\in\gamma}\tau\left(\chi\right) \ge \psi\left(|\gamma|\right) * \sum\nolimits_{\chi\in\gamma}\rho\left(\chi\right).$$

Note that if the metric is defined $\mu\left(\langle A,B\rangle\right) = |A-B|$ then the cardinality (norm) of the set can be expressed inversely $|S| = \mu\left(\langle S,\emptyset\rangle\right)$.

Assume that the inequalities with respect to π and ψ are always satisfied. Taking on such additional requirements as $\overline{\langle\{\sigma\},\lambda\rangle} = \left\langle 2^{\theta}, 2^{\theta}\times 2^{\theta}\right\rangle$ we are considering the following concepts in order to establish the basis of the model.

Introscalar product of sets is [8]:

$$is\left(\langle A,B,S\rangle\right) = |S-(S\cap(A-B))| - |S\cap(A-B)|\,;$$

$$is\left(\langle A,B,S\rangle\right) = |S| - 2*|(S\cap A)-(S\cap B)|\,;$$

$$is\left(\langle A,B\rangle\right) = is\left(\langle A,B,A\cup B\rangle\right).$$

Introcoscalar product of sets is:

$$ics\left(\langle A,B,S\rangle\right) = \pm 2*\sqrt{|S-(S\cap(A-B))| * |S\cap(A-B)|};$$

$$ics\left(\langle A,B\rangle\right) = ics\left(\langle A,B,A\cup B\rangle\right).$$

These concepts make it possible to define an analogue of trigonometric functions for sets and a formal analogue of the Euler formula:

$$icos\left(\langle A,B,S\rangle\right) = \frac{is\left(\langle A,B,S\rangle\right)}{|S|};$$

$$isin\left(\langle A,B,S\rangle\right) = \frac{ics\left(\langle A,B,S\rangle\right)}{|S|};$$

$$iexp\left(\langle A,B,S\rangle\right) = icos\left(\langle A,B,S\rangle\right) + i * isin\left(\langle A,B,S\rangle\right);$$

$$iexp\left(\langle A,B\rangle\right) = iexp\left(\langle A,B,A\cup B\rangle\right).$$

Two sets are subintroorthogonal if and only if:

$$iexp\left(\langle A,B,S\rangle\right) = \pm i;$$

$$iexp\left(\langle A,B\rangle\right) = \pm i.$$

Two sets are subintroorthogonal if and only if the square of the introscalar product is minimal:

$$is(\langle A,B,S\rangle)^2 \to \min;$$

$$is(\langle A,B\rangle)^2 \to \min.$$

A family of sets is called a introorthogonal basis if and only if any different of them are pairwise introorthogonal.

A family of sets is called a subintroorthogonal basis [8] iff any different of them are pairwise subintroorthogonal.

Thus, the subintroorthogonal basis can be chosen as the basis of the model (Fig. 16).

In the case of stronger restrictions it is required to additionally determine the maximum possible value of the area Γ.

The introduced models and concepts make it possible to transfer the obtained results to an distensible taxonomy, which has a distensible set of classes. They also provide an opportunity to simulate the adaptation of the model basis of an distensible taxonomy when new classes are added to it. It can also be used when distensible sets (kinds) are used instead of taxonomy classes.

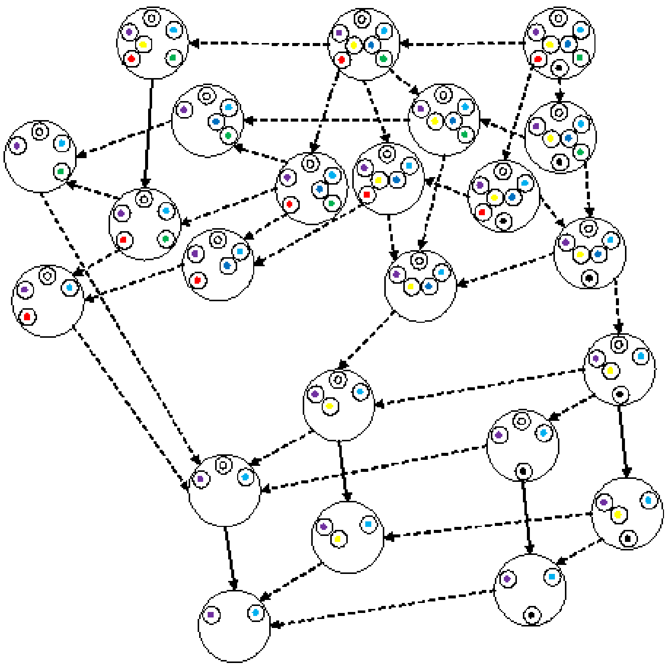


Fig. 16. Taxonomy states-transition diagram.

VII. Conclusions

The proposed models and approaches form the basis for solving problems in knowledge-driven systems aimed at ensuring interoperability and convergence of OSTIS ecosystem users and agents [69], [3], [9].

It seems promising to further study the properties of the semantic space and develop the OSTIS [5], [2] standard and technology based on the results obtained, including solving the problems of quality analysis and management of knowledge base and agents of intelligent computer systems.

References

[1] A.S. Narinyani. NE-faktory: netochnost' i nedoopredelennost' – razlichie i vzaimosvyaz' [Non-factors: inaccuracy and underdetermination – difference and interrelation]. Izv RAN (RAS). Ser. Teoriya i sistemy upravleniya 5, 2000. pp. 44—56.

[2] V.V. Golenkov. Otkrytyi proekt, napravlennyi na sozdanie tekhnologii komponentnogo proektirovaniya intellektual'nykh sistem [An open project aimed at creating a technology for the component design of intelligent systems], Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], 2013, pp. 55—78.

[3] V.P. Ivashenko. Modeli resheniya zadach v intellektual'nykh sistemakh. V 2 ch. Ch. 1 : Formal'nye modeli obrabotki informatsii i parallel'nye modeli resheniya zadach : ucheb.-metod. posobie [Models for solving problems in intelligent systems. In 2 parts, Part 1: Formal models of information processing and parallel models for solving problems: a tutorial] Minsk, BGUIR, 2020. 79 p.

[4] T.A. Gavrilova, V.F. Khoroshevsky. Bazy znanii intellektual'nykh sistem [Knowledge bases of intelligent systems], Saint Petersburg, Piter, 2001. 384 p.

[5] V.V. Golenkov, N.A. Gulyakina, D.V. Shunkevich. Otkrytaya tekhnologiya ontologicheskogo proektirovaniya, proizvodstva i ekspluatatsii semanticheski sovmestimykh gibridnykh intellektual'nykh komp'yuternykh system [Open technology for ontological design, production and operation of semantically compatible hybrid intelligent computer systems], Minsk, Bestprint, 2021. 690 p.

[6] Z. Pawlak. Rough sets. International Journal of Parallel Programming, 1982, vol. 11, no. 5, pp. 341—356.

[7] D. Dubois, H. Prade. Rough fuzzy sets and fuzzy rough sets International Journal of General Systems, 1990, vol. 17, no. (2-3), pp. 191-–209.

[8] V.P. Ivashenko. Modeli i algoritmy integratsii znanii na osnove odnorodnykh semanticheskikh setei (disc. na soiskanie stepeni kand. tekhn. nauk: 05.13.17) [Models and algorithms for knowledge integration based on homogeneous semantic networks (thesis for the degree of Candidate of Technical Sciences: 05.13.17)] , Minsk, BGUIR, 2014, 152 p.

[9] V.P. Ivashenko Operatsionnaya semantika mnogoagentnykh sistem obrabotki znanii [Operational semantics of multi-agent knowledge processing systems], Information Technologies and Systems, 2020, Minsk, BGUIR, pp. 78–79.

[10] V.P. Ivashenko. Modeli i algoritmy integratsii znanii na osnove odnorodnykh semanticheskikh setei [Models and algorithms for knowledge integration based on homogeneous semantic networks], Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], 2015, Minsk, BGUIR, pp. 111—132.

[11] V.P. Ivashenko Tekhnologiya razrabotki programmnykh komponentov intellektual'nykh sistem na osnove integratsionnoi platformy [Technology for the development of software components of intelligent systems based on an integration platform], Information Technologies and Systems, 2021, Minsk, BGUIR, pp. 84–85.

[12] V.P. Ivashenko. Ontologicheskaya model' prostranstvenno-vremennykh otnoshenii sobytii i yavlenii v protsessakh obrabotki znanii [Ontological model of the space-time relations of the event and the phenomenon in the processes of knowledge processing] 2017, vol. 5, no. 107, Minsk, BSUIR, pp. 13–17.

[13] V.P. Ivashenko. String processing model for knowledge-driven systems. Minsk, Doklady BGUIR, 2020, vol. 18, no. 6, pp. 33–40.

[14] Yu. Manin, M. Marcolli. Semantic spaces. Published, Location, 2016. 32 p. (arXiv)

[15] M.A. Perez, D.I. Spivak. Toward formalizing ologs, Preprint arXiv:1503.08326, 2015. 35 p.

[16] D.I. Spivak, R.E. Kent. Ologs: a categorical framework for knowledge representation. Preprint arXiv:1102.1889, 2011. 52 p.

[17] V.V. Golenkov, O.E. Eliseeva, V.P. Ivashenko. Predstavlenie i obrabotka znanii v parallel'nykh grafodinamicheskikh assotsiativnykh mashinak [Representation and processing of knowledge in parallel graphodynamic associative machines], Minsk, BGUIR, 2001. 412 p.

[18] H. Alt, M. Godau, Computing the Fr´echet distance between two polygonal curves, Int. J. Comput. Geom. Appl., 1995, vol. 5, pp. 75–91.

[19] E. Study. Kürzeste Wege im komplexen Gebiet. Mathematische Annalen (in German). Springer Science and Business Media LLC, 1905, vol. 60 no. 3, pp. 321–378.

[20] G. Rozenberg, A. Salomaa. Handbook of Formal Languages, Volume 1: Word, Language, Grammar. Verlag, Berlin, Heidelberg, Springer, 1997. 873 p.

[21] B. Smith. Metody i algoritmy vychislenii na strokakh [Computing Patterns in Strings], Moscow, OOO "I.D. Williams", 2006. 496 p.

[22] D.A. Pospelov. Situatsionnoe upravlenie: teoriya i praktika [Situational management: theory and practice], Moscow, Nauka, 1986. 288 p.

[23] A. Blass. A game semantics for linear logic. Annals of Pure and Applied Logic, 1992, vol. 56, pp. 183–220.

[24] J.H. Conway. On Numbers and Games. AK Peters/CRC Press, 2000. 242 p.

[25] G. Plotkin. Call-by-name, call-by-value and the lambda-calculus. Theoretical Computer Science, 1975, vol. 1, no. 2. pp. 125–159.

[26] D. Scott, Ch. Strachey. Toward a mathematical semantics for computer languages. Proceedings of the Symposium on Computers and Automata, Microwave Research Institute Symposia Series, Polytechnic Institute of Brooklyn Press, New York, 1971, vol. 21, pp. 19—46.

[27] F. Daneš. On Prague school functionalism in linguistics. Functionalism in Linguistics, 1987, pp. 3–38.

[28] A. R. D. Mathias. Unordered pairs in the set theory of Bourbaki 1949, Archiv der Mathematik, 2010, vol. 94, pp. 1–10.

[29] W. Quine. On ordered pairs. Journal of Symbolic Logic, 1945, vol. 10, no. 3, pp. 95—96.

[30] J. Heijenoort. From Frege to Gödel. A source book in mathematical logic, 1879–1931. Cambridge, Mass., Harvard University Press, 1967, 664 p.

[31] C. Kuratowski. Sur la notion de l'ordre dans la Théorie des Ensembles. Fundamenta Mathematicae, 1921, vol. 2, no. 1,pp. 161–171.

[32] A.P. Morse. A Theory of Sets. Academic Press, 1965, 130 p.

[33] A. Levy. Basic Set Theory, Springer-Verlag, 1979, 391 p.

[34] K.J. Devlin. Fundamentals of contemporary set theory. Universitext. Springer-Verlag, 1979, 182 p.

[35] J.M. Lee. Introduction to Topological Manifolds, Springer, 2011, 452 p.

[36] J. McCarthy. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. Communications of ACM, New York, 1960, vol. 3, no. 4, pp. 184–195.

[37] G. W. F. Hegel. Nauka logiki [Science of Logic]: v 3-h t. T.1, Moscow, "Mysl'", 1970. 501 p.

[38] Platon. Sobr. soch. v 4-kh tomakh. [Collected works in 4 volumes.]: T. 3, Moscow, "Mysl'", 1994. 654 p.

[39] J. Cottingham, René Descartes: Meditations on First Philosophy: With Selections from the Objections and Replies, Cambridge, Cambridge University Press, 2015. 282 p.

[40] R.T.W. Arthur. Leibniz on Time, Space, and Relativity, Oxford University Press, 2022, 432 p.

[41] D. Hilbert. On the infinite. In Philosophy of Mathematics (1984), Selected Readings, Cambridge, Cambridge University Press, 2012, pp. 183–201.

[42] D. Bohm and B.J. Hiley. The Undivided Universe: An Ontological Interpretation of Quantum Theory, London, Routledge, 1993. xii + 397 p.

[43] D. Bohm. Wholeness and the Implicate Order, London, Routledge, 2002. 284 p.

[44] V.V. Nalimov, Zh.A. Drogalina. Real'nost' nereal'nogo. Veroyatnostnaya model' bessoznatel'nogo [The reality of the unreal. Probabilistic model of the unconscious], Moscow, Mir idei, AO Akron, 1995. 432 p.

[45] V.V. Nalimov. Spontannost' soznaniya. Veroyatnostnaya teoriya smyslov i smyslovaya arkhitektonika lichnosti [Spontaneity of consciousness. Probabilistic theory of meanings and semantic architectonics of personality], Moscow, Prometei, 1989. 287 p.

[46] V.V. Nalimov. Veroyatnostnaya model' yazyka. O sootnoshenii estestvennykh i iskusstvennykh yazykov [Probabilistic model of language. On the relationship between natural and artificial languages] Moscow, Nauka, 1979. 304 p.

[47] I.N. Taganov, Yu.I. Babenko. Antivremya i antiprostranstvo [Anti-time and anti-space], Saint Petersburg, RAN, 2016. 200 p.

[48] S. Roman, Advanced Linear Algebra, Graduate Texts in Mathematics, vol. 135 (2nd ed.), Berlin, New York, 2005. 488 p.

[49] M.A. Armstrong, Basic Topology [1979], Undergraduate Texts in Mathematics, Springer, 1983. 263 p.

[50] L. Collatz. Functional Analysis and Numerical Mathematics, New York, San Francisco, London, Academic Press, 1966. xx + 473 p.

[51] J. Heinonen. Lectures on analysis on metric spaces, New York, Springer, 2001. X+141 p.

[52] A.A. Adronov, A.A. Vitt, S.E. Haikin. Teoriya kolebanii (2-e izd.) [Oscillation theory (2nd ed.)], Moscow, Nauka. 1981. 586 p.

[53] GOST 7427-76. Geometricheskaya optika. Terminy, opredeleniya i bukvennye oboznacheniya. S izmeneniem №1, utverzhdennym v iyule 1982 g. [GOST 7427-76. Geometric optics. Terms, definitions and letter designations. With change No. 1, approved in July 1982]. Moscow, Izdatel’stvo standartov, 1988. 19 p.

[54] H.H. Schaefer, M.P. Wolff. Topological Vector Spaces, New York, Springer New York Imprint Springer, 1999. 361 p.

[55] I. Mel’ˇcuk. Language: from Meaning to Text. Ed. by D. Beck. Moscow & Boston, 2016.

[56] J. Harris. Algebraic Geometry: A First Course, New York, Springer, 1992. 330 p.

[57] A. Kostrikin, Yu. Manin. Linear Algebra and Geometry, Gordon and Breach Science Publishers, 1997. 320 p.

[58] W. Lowe. Towards a theory of semantic space, in Proceedings of the 23rd Conference of the Cognitive Science Society, 2001, pp. 576–581.

[59] Yu. I. Manin. Zipf’s law and L. Levin’s probability distributions. Functional Analysis and its Applications, 2014, vol. 48, no. 2. Preprint arXiv:1301.0427.

[60] V.V. Martynov. V tsentre soznaniya cheloveka [At the center of human consciousness], Minsk, BGU, 2009. 272 p.

[61] V.V. Martynov. Yazyk v prostranstve i vremeni: k probleme glottogeneza slavyan (2-e izd.) [Language in space and time: to the problem of glottogenesis of the Slavs (2nd ed.)], Moscow, Editorial URSS, 2004. 106 p.

[62] A.N. Gordey. Teoriya avtomaticheskogo porozhdeniya arkhitektury znanii (TAPAZ-2) i dal’neishaya minimizatsiya semanticheskikh ischislenii [The theory of automatic generation of knowledge architecture (TAPAZ-2) and further minimization of semantic calculus]. Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems], 2014, Minsk, BGUIR. pp. 49–64.

[63] C. Marijuan. Finite topologies and digraphs. Proyecciones (Antofagasta), 2010, vol. 29, pp. 291–307.

[64] R. Penrose. The Road to Reality, Oxford, Oxford University Press, 2004. 1094 p.

[65] A.K. Guts. Khronogeometriya. Aksiomaticheskaya teoriya otnositel’nosti [Chronogeometry. Axiomatic theory of relativity], Omsk, OOO “UniPak”, 2008. 340 p.

[66] V. Ivashenko, N. Zotov, M. Orlov. Semantic Logging of Repeating Events in a Forward Branching Time model. Pattern Recognition and Information Processing (PRIP’2021), United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, 2021, pp. 149–152.

[67] M. Bansal, D. Burkett, G. de Melo, Gerard, D. Klein, Dan. Structured Learning for Taxonomy Induction with Belief Propagation. 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 – Proceedings of the Conference, 2015, vol. 1, pp. 1041–1051.

[68] Yu.I. Zhuravlyov. Ob algebraicheskom podkhode k resheniyu zadach raspoznavaniya ili klassifikatsii [On an algebraic approach to solving problems of recognition or classification]. Problemy kibernetiki, 1978, vol. 33. pp. 5–68.

[69] V.V. Golenkov, D.V. Shunkevich. Agentno-orientirovannye modeli, metodika i sredstva razrabotki sovmestimykh reshatelei zadach intellektual’nykh sistem [Agent-based models, methodology and tools for developing compatible problem solvers for intelligent systems]. Programmnye produkty i sistemy, 2020, vol. 33, no. 3. pp. 404–412.

[70] J.A. Fodor. The Language Of Thought. Crowell Press, 1975. 214 p.

[71] A.N. Kolmogorov. Problems of Probability Theory. Theory of Probability & Its Applications, 1994, vol. 38, no. 2, pp. 177–178.

Универсальный язык смыслового представления знаний и смысловое пространство

Ивашенко В.П.

Статья рассматривает модели и средства, обеспечивающие унифицированное представление знаний и их интеграцию в рамках «смыслового пространства». Для этого вводится понятие «обобщённого формального языка», позволяющего выявить с целью анализа взаимоотношение формальных языков и известных языков представления знаний, включая семантические сети.

На основе этого анализа уточняется семантика языков модели унифицированного представления знаний, вводится язык, являющийся основой стандарта для технологии разработки интеллектуальных систем, и даётся концепция «смыслового пространства», ориентированного использование в целях оценки качества интеллектуальных компьютерных систем в рамках технологии OSTIS. Рассматриваются прикладные задачи на основе предложенных моделей и дальнейшие перспективы развития технологии и её компонентов.

Received 11.11.2022

Family of external languages of next-generation computer systems, close to the language of the internal semantic representation of knowledge

Alexandra Zhmyrko
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: aleksashazh@gmail.com

***Abstract*—In the article, the concepts of external and internal languages of next-generation intelligent computer systems are considered. External languages of knowledge representation within the OSTIS Technology are described, namely SCg-code, SCs-code, SCn-code. For each of the external languages, its syntax and denotational semantics are considered in detail.**

***Keywords*—next-generation intelligent computer system, external language, internal language, OSTIS, SC-code, SCg-code, SCs-code, SCn-code**

I. Introduction

At the current stage of information technologies development, the problem of ensuring semantic interoperability of computer systems and their components is the most important and significant. To solve this problem it is necessary to move from traditional computer systems and modern intelligent computer systems to computer systems based on the semantic representation of information (next-generation intelligent computer systems).

Such systems have a high level of learnability, i.e. the ability to rapidly acquire new and improve already immersed knowledge and skills, while having no limitations on the type of knowledge and skills gained, improved, and shared. The components of such systems have a high degree of compatibility, which virtually eliminates the duplication of engineering solutions and makes it possible to significantly accelerate the development of computer systems based on the semantic representation of information through a constantly expanding library of reusable and compatible components.

Next-generation intelligent computer systems require an internal language to represent information in a meaningful way. By internal language is meant the language used by a system to represent the information stored in its memory [1].

For operation of next-generation intelligent computer systems, except for a method of abstract internal representation of knowledge bases, methods of external representation of abstract texts convenient for users and used at registration of initial texts of knowledge bases of the specified intelligent computer systems and initial texts of fragments of these bases, as well as used for display of various fragments of knowledge bases on user request, are required [2].

All basic external formal languages are variants for the external representation of the texts of the internal language of the system. Such languages are universal and therefore semantically equivalent.

For any language, syntactic rules (rules for constructing information constructions of such a language) and semantic rules (denotational semantics – rules for relating to those entities and configurations of entities that are described (reflected) by the specified sign constructions) must be specified.

A next-generation intelligent system must be able to visualise certain information in different ways. Each visualisation option requires the design and development of visualisation languages and tools to translate these languages from the system internal representation language into an external language. At the moment, the **lack of a universal mechanism for describing** external languages, translators for them, and their “seamless” integration into the system remains problematic.

In the article, a family of external languages of next-generation intelligent computer systems close to the language of internal knowledge representation on the example of ostis-systems is considered. For each of the external languages, its syntax, denotational semantics, and hierarchical family of semantically equivalent sublanguages are described in detail.

II. State of art

Knowledge representation languages are frequently difficult to understand, particularly for those who is not trained in formal logic. They are in common used to describe domains ranging from biology to finance. These languages are typically used by both computer scientists and domain experts [3].

It is often said that a picture is worth a thousand words. That is true of sketches, diagrams, and graphs used in various fields of knowledge. Conceptual maps are widely used in education to represent and clarify complex relations between concepts. Flowcharts serve as graphical representations of procedural knowledge or algorithms. Decision trees are another form of representation used in various fields, particularly in decision-making or expert systems.

All these representation methods are useful at an informal level, as thinking aids and tools for the communication of ideas, but they have limitations. One is the imprecise meaning of the links in the model. Non-typed arrows can mean many things, sometimes within the same graph. Another problem is the ambiguity around the type of entities. Objects, actions on objects, and propositions of properties about them are all mixed-up, which make graph interpretation a fuzzy and risky business.

Another difficulty is to combine more than one representation in the same model. For example, concepts used in procedural flowcharts as entry, intermediate, or terminal objects could be given a more precise meaning by developing them in conceptual maps as sub-models of the procedure. The same is true of procedures represented in conceptual models that could be developed as procedural sub-models described by flowcharts, combined or not with decision trees. In software engineering, many graphic representation formalisms have been or are used such as EntityRelationship models [4], Conceptual Graphs [5], Object modelling technique (OMT) [6], KADS [7], or the Unified Modeling Language (UML) [8]. These representation systems have been built for the analysis and architectural design of complex information systems. The most recent ones require the usage of up to eight different kinds of model so the connections between them become rapidly hard to follow without considerable expertise.

Graphic representation system should be both simple enough to be used by educational specialists who are not computer scientists in general, be general and powerful enough to represent the components of computer-based educational environments and their relations.

Graphic. The benefits of graphical cognitive modelling have been eloquently summarized by Ausubel [9], Dansereau [10], and Jonassen [11]. Graphs illustrate relations among components of complex phenomena. They uncover the complexity of actor interactions. They facilitate the communication about the reality studied. They favour the global comprehension of studied phenomena. They help grasp the structure of related ideas by minimizing the usage of ambiguous natural language texts. As an example, entity-relation graphs reduce ambiguity compared to a natural language description but some remain on the interpretation of the terms written on the connections or nodes. Ambiguity can be reduced further by the usage of standardized typed objects and typed connections.

User-friendliness. Not all graphic modeling languages are user-friendly. A good counter-example is UML. The large number models and symbols require considerable expertise for the interpretation and construction of the model of a system. Furthermore, each type of model captures a different viewpoint on the information, and it is impossible to mix them in the same graph to provide a global view of a subject domain. The representation system must be easy to use without technical or scientific mastery after a short period of initiation. Dansereau and Holley [12] have studied experimentally the usage of different sets of graphic symbols by learners. Their results show that typed connections are preferred by the majority of learner, as long as there are neither too few nor too many types of connections and they are clearly differentiated with well-defined meanings.

General. Generality means that the representation language should have the capacity to represent, with a relatively small number of objects and connection categories, knowledge in very different subject domains, at various levels of granularity and precision. It should be possible to represent simple models such as a multiplication table, up to complex models such as multi-actor workflows, rule-based systems, methods, and theories. It should also be possible to offer equivalent representations to commonly used graphs such as conceptual maps, semantic networks, flowcharts, decision trees, or cause/effect diagrams.

Formalizable. The graphic language should be upward compatible from informal graphs, up to semiformal and totally unambiguous formal models. At the informal level, an integrated representation framework facilitates thought organization and communication between humans about the knowledge as the graphic representation model evolves. Here, the process is more important than the result. At the other hand, the graphic language offers more constrained elements to produced totally unambiguous descriptions that can be exported to set of symbols, such as an XML file, that can be processed by computer agents. Here, the model is more important than the process.

Declarative. Graphic language can be procedural or declarative. Procedural graphic languages have been built in the past, extending flowcharts to promote graphical programming that produces code directly. However, declarative language is, firstly, easier for a human to declare the components of their knowledge than to describe also the way it should be processed. In expert systems, for example, the execution instructions are not wired in the program but externalized and made visible in a knowledge base on which a general inference engine proceeds. Secondly, the same model can be used for many different applications not necessarily the one for which the processing has been planned in a procedural program. This is done by querying the model using an inference engine, in a Prolog-like manner. Thirdly, the processing knowledge itself can be given declaratively, so that higher order metaknowledge can be also singled-out. This idea is similar to structural analysis [13] and is exactly the way we should see the relation between generic skills and specific domain knowledge in a competency, as meta-knowledge given declaratively, applied to domain knowledge. For example, rules for diagnosing a component-based system applied to models describing a car, a software, or a learning environment provide a good way to represent generic skills and competencies.

Standardized. Standardization is an important property to enlarge knowledge communication and use between humans and/or software agents. At the informal level, each model constructed by a human must be interpretable by another human.

Computable. Computability is a step beyond standardization. The graphic model can not only receive a non-ambiguous formal representation that can be processed by computer agents, but this formal representation is complete (all conclusions are guaranteed to be computable) and decidable (all computations will finish in finite time) [14].

Thus, knowledge representation languages in next-generation intelligent computer systems must comply with the above properties.

III. Proposed approach

To solve the problem of integrating new external languages of knowledge representation into the system, it is proposed to describe external languages on the basis of ontologies. As already mentioned, each language is defined by its syntax and denotational semantics, which can be written in an ontological way, which will allow universalising and docking these languages with each other, creating tools for visualising and understanding the languages, making them more universal.

The OSTIS Technology, a next-generation technology for intelligent computer system design, is proposed as a tool to implement the specified approach.

The advantages of the OSTIS Technology:

- at the heart of the OSTIS Technology, there is an *SC-code*, which allows any information to be represented in a unified (same) way, making the proposed approach universal and suitable for any class of intelligent system;
- the OSTIS Technology and the *SC-code* in particular can be easily integrated with any modern technology, allowing the proposed approach to be applied to a large number of already developed intelligent systems;
- the *SC-code* allows storing and describing in the knowledge base of the *ostis-system* any external (foreign) information in relation to the *SC-code* in the form of internal ostis-system files. Thus, the knowledge base of the training subsystem can contain explicitly fragments of already existing documentation for the system, represented in any form;
- the OSTIS Technology has already developed models for ostis-system knowledge bases, ostis-system problem solvers, and ostis-system user interfaces, assuming their complete description in the system knowledge base. Thus, for ostis-systems, the proposed approach to training end-users and developers is much easier to implement and provides additional benefits;
- one of the main principles of the OSTIS Technology is to ensure the flexibility (modifiability) of the systems developed on its basis. Thus, the usage of the OSTIS Technology will enable the evolution of the intelligent learning subsystem itself [2].

The systems developed on the basis of the OSTIS Technology are called *ostis-systems*. The universality of the *SC-code* is ensured by the fact that the elements of the *SC-code* texts can be signs of described entities of any kind, including connection signs between the described entities and/or their signs. Accordingly, the texts of the *SC-code* are graph structures of an extended form, in which the characters of the described connections can connect not only the vertices (nodes) of the graph structure but also the characters of other connections.

The *SC-code* is an abstract language, i.e. a language for which the way, in which the characters (syntactically elementary fragments) that make up the texts of this language are represented, is not specified but only the alphabet of these characters, i.e. the family of character classes considered syntactically equivalent to each other, is specified.

Each abstract language can be assigned a whole family of real languages providing isomorphic real representation of texts of the specified abstract language by specifying ways of representation (representation, coding) of symbols included in these texts, as well as by specifying rules for establishing syntactic equivalence of these symbols. Obviously, in all other respects, the syntax and denotational semantics of the mentioned real languages completely coincides with the syntax and denotational semantics of the corresponding abstract language.

Every intelligent system operates with a knowledge base in an internal language, and the dialog takes place as an exchange of messages between the user and the system. For such a dialog to take place, a fragment of the knowledge base must be displayed into an external form. Such a form can be either universal or specialized.

Within the *OSTIS Technology*, three universal external knowledge representation languages are proposed:

- the **SCg-code** – one possible way of visually representing *SC-texts*. The basic principle behind the *SCg-code* is that each *sc-element* is matched with an *sc.g-element* (graphical representation);

- the **SCs-code** – string (linear) representation of the *SC-code*, designed to represent sc-graphs (texts of *SC-code*) as sequences of characters;
- the **SCn-code** – string non-linear variant for representation of the *SC-code*. The *SCn-code* is intended to represent *sc-graphs* as formatted sequences of characters according to predefined rules, within which basic hypermedia such as graphical images can be used, as well as means of navigation between parts of *sc.n-texts* [15].

Each of these languages meets the requirements for universal languages of knowledge representation and allows the user to choose the most convenient variant of visual representation of any subject domain. In addition, each of these languages has the unique property of being able to be described **in the same language**, being able to **be translated from one to the other**. Thus, it is proposed to use *SCg-code, SCs-code, and SCn-code* as knowledge representation languages, whose syntax and denotational semantics will be considered within this article.

IV. Internal language of the *ostis-system* – an SC-code

SCg-code, SCs-code, and SCn-code are sub-languages of the *SC-code*, which define the syntactic, semantic, and functional principles of memory organisation in next-generation computers focused on the implementation of next-generation intelligent computer systems.

The *SC-code* texts (sc-texts) are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which in turn can be *sc-arcs* or *sc-rules*, depending on their orientation). The *Alphabet of the SC-code* consists of five basic elements, on the basis of which SC-code constructions of any complexity are built, including the introduction of more specific types of sc-elements (e.g. new concepts). A detailed description of the *SC-code* can be found in the standard [16].

The signs (designations) of all entities described in *sc-texts* (SC-code texts) are represented as syntactically elementary (atomic) fragments of *sc-texts* and therefore have no internal structure, not consisting of simpler text fragments, such as names (terms), which represent signs of described entities in familiar languages and consist of letters.

Names (terms), natural language texts, and other information constructions which are not *sc-texts* can be included in *sc-text* but only as files described (specified) by *sc-texts*. Thus, a knowledge base of an intelligent computer system based on the *SC-code* can include names (terms) denoting some describable entities and represented by corresponding files.

Each *sc-element* will be called an internal designator of some entity, and the name of this entity represented by the corresponding file will be called an external identifier (external designator) of this entity. Each named (identifiable) sc-element is connected by an arc of membership to the "to be an external identifier*" relation with a node whose content is an identifier file (in particular, a name) denoting the same entity as the above sc-element. The external identifier can be a name (term) but also a hieroglyph, a pictogram, a spoken name, a gesture. It should be emphasized that external identifiers of described entities in an intelligent computer system based on the *SC-code* are used only:

- to analyse information coming into this system from various sources and to input (understand and immerse) this information into the knowledge base;
- to synthesise different messages addressed to different subjects (including users).

V. Identification of sc-elements

External *sc-element* identifiers (or, for short, *sc-identifiers*) are necessary for the *ostis-system* to exchange information with other *ostis-systems* or with its users. In order to represent its knowledge base, to solve various problems related to analysis of the current state and evolution of its knowledge base, problems related to analysis of the current state (current situations) of the environment, making appropriate decisions (purposes), and organising appropriate actions to implement the decisions made (to achieve the purposes), the *ostis- system* does not need any external identifiers (in particular names) corresponding to *sc-elements*.

However, in order to understand messages received from other subjects (which for the *ostis-system* means to construct the *sc-text* semantically equivalent to the received message) and to analyze messages transmitted to other subjects (which for the *ostis-system* means synthesizing an external text that is semantically equivalent to a given *sc-text* and meeting some additional requirements, such as an emotional one). The *ostis-system* needs to know how characters that are synonymous with sc-elements which are or could be stored in the knowledge base of the *ostis-system* are represented in the message being received or transmitted.

The external identifiers of *sc-elements* are most often the names (terms) of the corresponding (denoted) entities, represented by single words or phrases in various natural languages, however, hieroglyphs, conventions, pictograms can also be used.

In general, an *sc-element* can correspond to several synonymous names in different natural languages. Moreover, an *sc-element* can correspond to several synonymous names in the same natural language. In this case, one of these names is declared as the main external identifier for the corresponding *sc-element* and the corresponding natural language. The main requirement for such external identifiers is that there are no synonyms as well as homonyms within the set of basic external identifiers of *sc-elements* for each natural language.

Each external *sc-element* identifier used by the *ostis-system* can be described (represented) in its memory as an internal *ostis-system* file, i.e. as an electronic image of all possible occurrences of this *external identifier* in all possible external texts of the corresponding external language. In some cases, an explicit representation in memory is not required, e.g. in the case of *non-translatable sc-identifiers*.

Next, let us consider the external universal languages of knowledge representation.

VI. SCg-code. Alphabet of the SCg-code and denotational semantics

An *SCg-code* is a way of visualising *sc-texts* (SC-code information constructions) as drawings of these abstract structures. We emphasize that an abstract graph structure and its drawing (graphical representation) are not the same thing even if they are isomorphic to each other.

We consider the *SCg-code* as an combination of the *SCg-code Core*, which provides an isomorphic graphical representation of any *sc-text* and several extensions to this core that provide increased compactness and "readability" of *SCg-code (sc.g-texts)* texts.

The main purpose of the *SCg-code* is to have a clear syntactic graphic representation of *sc.g-elements*, allowing the classes of *sc.g-elements* to be easily identified and distinguished, such as:

- *sc.g-constants* (signs of constant entities) and sc.g-variables (images of variables whose values are the corresponding sc-elements);
- *sc.g-variables* whose values are sc-constants and sc.g-variables whose values are sc-variables;
- signs of permanent (stable) entities and signs of temporal (unstable, temporary existing, situational) entities;
- *sc.g-connectors* (binary characters) and *sc.g-elements* that are not sc.g-connectors;
- *non-oriented sc.g connectors* (sc.g edges) and oriented (sc.g arcs);
- *sc.g-arcs of membership* and sc.g-arcs that are not such;
- *sc.g-arc* of positive membership, negative membership, and fuzzy membership.

Figure 1 is the element list for the Alphabet of the SCg-code.

This list is created in the form of *sc.g-text* and is a representation for examples of all put types of *sc.g-elements* (one example of each type). At the same time, the specified examples of *sc.g-elements* are divided into five groups (SCg-text. Alphabet of the SCg-code). The first group (top row) includes *sc.g-element* for which the constancy and consistency of the entities they denote requires further specification. The remaining four groups of *sc.g-elements* are similar to each other and include, respectively:

- signs of constant permanent entities;
- signs of constant temporal entities;
- images of sc-variables whose values or whose value values (in case the values of the variables are variables) are the signs of constant permanent entities;
- images of sc-variables whose values or whose value values (in case the values of the variables are variables) are the signs of constant temporal entities.

A special point of the *SCg-code* is the representation of *sc-elements*, which are designations of the membership pair* by explicitly using this semantically distinguishable class of *sc-elements*. This *sc.g-element* is used when we need to represent an *sc-arc* that is known to be a designation of the membership pair*, but it is not known whether it is constant or variable, permanent or temporal, positive, negative, or fuzzy.

In addition to the *sc.g-elements* listed in Figure 1, the Alphabet of the SCg-code also includes the following *sc.g-elements*:

- external sc-element identifiers that are identical (attributed) to the corresponding *sc.g-elements*;
- sc.g-contours, each of which is a sign of some sc-text (a structure consisting of sc-elements). Each such sc-text can be:
  - either a constant permanent structure;
  - a constant temporal structure (situation);
  - or a variable structure whose values are permanent structures of an isomorphic configuration;
  - or a variable structure whose values are temporal structures (situations) of an isomorphic configuration.
- enlarged sc.g-frames that are image limiters for the various files stored in the *ostis-system* memory;
- sc.g-buses, which are designations of the same entities as their incident *sc.g-elements*.

Let us note also that, in addition to all the above elements of the Alphabet of the SCg-code, each of which has quite specific denotational semantics, a number of "smaller" syntactic objects need to be introduced to formalise the SCg-code syntax, e.g:

- incidence points of sc.g-connectors with sc.g-nodes, with other sc.g-connectors, with sc.g-contours, with sc.g-frames;
- sc.g-bus incidence points;
- salient points of linear *sc.g-elements* (sc.g-connectors, sc.g-contours, sc.g-frames, sc.g-buses).

Within the SCg-code, the SCg-code Core and its extensions are allocated. The Alphabet of SCg-code Core is an alphabet of *sc.g-elements* graphically represented by sc-elements. The Alphabet of the SCg-code Core is mutually unambiguous with the Alphabet of the SC-code.

The denotational semantics of the SCg-code Core correspond to the denotational semantics of the SC-code. This is demonstrated in Figure 2.

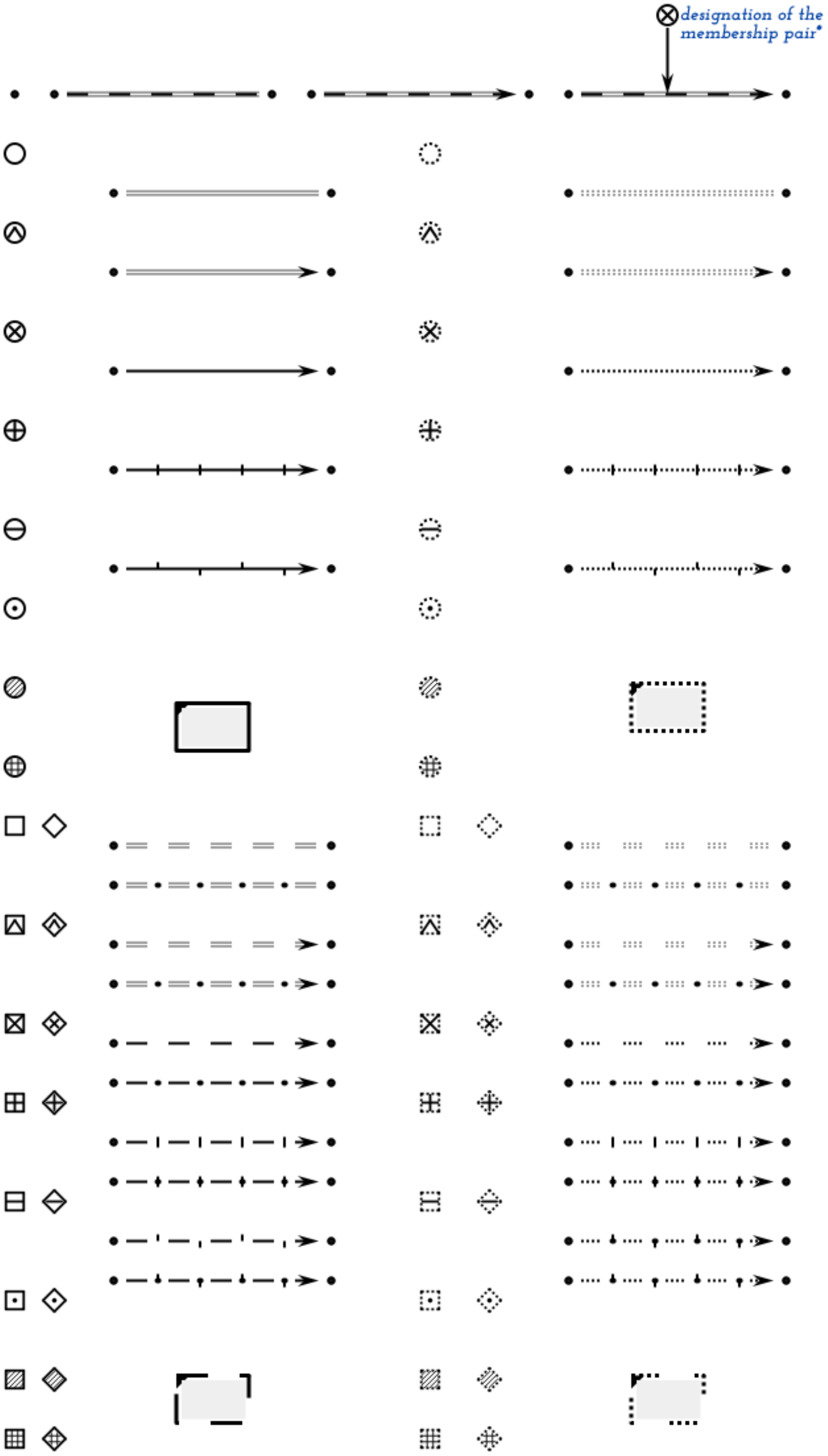


Figure 1. Elements of the Alphabet of the SCg-code

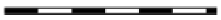
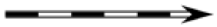
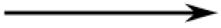
Alphabet of the SC-code	Alphabet of the SC-code Core	sc.g-element image
sc-node of a common type	sc.g-node of a common type	●
sc-edge of a common type	sc.g-edge of a common type	
sc-arc of a common type	sc.g-arcof a common type	
basic sc-arc	basic sc.g-arc	
internal ostis-system file	content sc.g-node	□

Figure 2. Denotational semantics of the SCg-code Core

The Alphabet of the SCg-code Core is represented by the following elements:

- an *sc.g-node of a common type* – an *sc.g-element* which is a graphical representation of the sc-node of a common type. All sc-nodes, which are not file signs, in the text (construction) of the SCg-code Core, are represented as small black circles of the same diameter, which we denote by 'd' and the exact value of which depends on the scale of sc.g-text;
- an *sc.g-edge of a common type* – an *sc.g-element*, which is a graphical representation of the sc-edge of a common type. Each sc-edge in the SCg-code Core is represented as a wide line with alternating solid-filled and non-filled fragments that have no self-intersections and an overall weight of about 0.7 d;
- an *sc.g-arc of a common type* – an *sc.g-element*, which is a graphical representation of the sc-arc of a common type. Each sc-arc in the SCg-code Core is represented as a wide line with alternating solid-filled and non-filled fragments, with no self-intersections, having an overall weight of about 0.7 d and having an arrow at one end of this line;
- a *basic sc.g-arc* – an *sc.g-element*, which is a graphical representation of the basic sc-arc. Each underlying sc-arc in the SCg-code Core is represented as an arbitrary shaped line without self-intersections, having a weight of 0.4 d and an arrow at one of its ends;
- an *internal ostis-file* – an *sc-node*, which is the sign of the internal *ostis-system* file;
- an *sc.g-node with contents* – an *sc.g-node*, which is the sign of the internal *ostis-system* file, an *sc.g-frame*;
- an *sc.g-frame* is always a rectangle, the maximum size of which is not limited, but the minimum size is fixed and corresponds to the *sc.g-frame*, inside which the file it designated is not displayed. Each sc-node in the *sc-text* that has contents is represented as an arbitrarily sized rectangle with a line weight of 0.6 d in the *SCg-code* Core. Inside this rectangle, it is possible to see the file indicated by the depicted *sc-node*. If there is no need to represent the file itself in the text, the *sc-node* denoting such a file is represented in *sc.g-text* as a rectangle with sides 2d vertically and 4d horizontally.

It is difficult to believe at once that such a simple alphabet can be used to build a convenient and versatile graph language. Besides, we should not be alarmed by simplicity of the alphabet because mankind has a great experience of coding, storing in memory, and transferring through communication channels the most various information resources using the alphabet consisting of only two classes of elements – ones and zeros. We are talking about a fundamentally different (graphical) way of encoding information in computer systems, but we try to reduce this encoding to a simple enough alphabet at least in order not to artificially complicate the problem of creating next-generation computers based on information encoding method. Extensions of the *SCg-code* Core will be considered as directions of transition from the *SCg-code* Core texts to more compact texts. However, since it leads to complication of the Syntax of the *SCg-code* and, first of all, to expansion of the *Alphabet of the SCg-code*, it is necessary to make such extensions reasonably taking into account frequency of occurrence within knowledge bases of *ostis-systems* of corresponding fragments.

VII. SCs-code. Alphabet of the SCs-code and denotational semantics

An *SCs-code* is a language of linear knowledge representation of *ostis-systems*. A set of linear texts (sc.s-texts), each consisting of sentences (*sc.s-sentences*) separated from each other by a double semicolon (separator of *sc.s-sentences*). In this case, the sc.s-sentence is a sequence of *sc-identifiers* which are the names of the described entities and are separated from each other by different *sc.s-separators* and *sc.s-delimiters*.

The *Alphabet of the SCs-code* is based on modern commonly used character sets, which simplifies the development of tools for working with *sc.s-texts* using modern technologies.

The *sc.s-texts*, as well as the texts of any other languages which are variants of the external representation of the *SC-code* texts, can include various files, including natural language files or even files containing other sc.s-texts. In general, such files can use a variety of characters, so we will assume that these characters are not included in the *Alphabet of the SCs-code*.

The alphabet of symbols used in *sc.s-separators* consists of: space, semicolon, colon, round marker, and equality sign.

The alphabet of symbols used in *sc.s-separators* displaying the incidence relation of sc-elements consists of: “<”, “>”, “|”, “-”.

The basic alphabet of characters used in *sc.s-connectors* consists of: “∼”, underscore sign, equality sign, colon, ‘<”, “>”, “|”, “-”, “/”.

The extended alphabet of symbols used in *sc.s-connectors* consists of “∈”, “∋”, “⊆”, “⊇”, “⊂”, “⊃”, “≤”, “≥”, “⇐”, “⇒”, “←”, “→”, “⇔”.

Both in the *Basic* and the *Extended* Alphabets of *sc.s-connectors*, the following common features to characterize the type of *sc-connector* being represented are used:

- an underscore as an image feature of the *sc-connectors* variables (one underscore for *sc-connectors* that are primary *sc-variables*, two underscores for *sc-connectors* that are secondary *sc-variables (sc-meta-variables)*);
- a vertical line “|” as an image feature of *negative sc-arcs of membership*;
- slash “/” as an image feature of *fuzzy sc-arcs of membership*;
- tilde “∼” as an image sign of *temporal sc-arcs of membership*.

To simplify the process of developing knowledge base source texts using the *SCs-code* and creating corresponding tools, two character alphabets are introduced. The basic alphabet of characters used in *sc.s-connectors* includes only the characters included in the portable character set and available on a standard modern keyboard. Thus, to develop the source code of knowledge bases using only the Basic alphabet of symbols used in *sc.s-connector*, a normal text editor is sufficient. The extended alphabet of characters used in *sc.s-connectors* also includes additional characters that make *sc.s-texts* (and *sc.n-texts*) more readable and clear. To visualize and develop sc.s-texts using the extended alphabet, specialized tools are required.

The alphabet of symbols used in *sc.s-delimiters* consists of: “(”, “)”, “*”.

The alphabet of symbols used in ambiguous *sc.s-images of sc-nodes* consists of: “{”, “}”, “-”, “!”, “ [”, “] .

Important elements of the *SCs-code* are the *sc.s-separator* and the *sc.s-delimiter*.

An *sc.s-separator* is a separator used in *sc.s-texts*. The *sc.s-separator* splits into:

- an *sc.s-separator* used to structure *sc.s-sentences*.
  - it separates the *sc-identifier of a binary relation* and the second component of one of its connectives, in case the specified binary relation and its connective are connected by a *constant sc-arc of membership*. It is represented as a colon.
  - Separates the *sc-identifier of a binary relation* and the second component of one of its connectives, in case the specified binary relation and its connective are connected by a *variable sc-arc of membership*. It is represented as a double colon.
- *sc.s-separator* of *sc.s-sentences* is represented as a double semicolon.

sc.s-delimiter is represented as: (![(∗]!∪![∗)]!)

The parentheses with an asterisk limit attached *sc.s-sentences*, which, in turn, may have other attached *sc.s-sentences* in their structure.

There is also an *sc.s-connector*. The typology of *sc.s-connectors* is fully consistent with that of *sc.g-connectors* and even more so with *sc-connectors*, since it takes into account the well-established tradition of representing the connectives of a number of specific relations. The following *sc.s-connectors* are distinguished:

- an *oriented sc.s-connector*,
- a *non-oriented sc.s-connector*;
- an *sc.s-connector*, corresponding to the *sc.g-arc of membership*,
- an *sc.s-connector* corresponding to a *sc.g-connector* that is not an *sc.g-arc*.

The set of *sc-elements* has a binary oriented sc-element incidence relation, as well as a subset of this relation – the incidence relation of incoming *sc-arcs*, each pair of which relates the sc-arc to the sc-element it is a part of. In the *SC-code*, *sc-connectors* can connect not only an *sc-node* with *sc-nodes* but also an *sc-node* with an *sc-connector* and even an *sc-connector* with an *sc-connector*. In the latter case, specifying the incidence of sc-connectors, it is necessary to specify which of them is connecting and which is connectable. Therefore, the incidence relation specified on the set of sc-elements is oriented. The first component of the pair of this relation is the connecting *sc-connector* and the second component is the connecting *sc-element*. Obviously, the connecting *sc-element* is always an *sc-connector* and the *sc-node* can only be connectable.

The *sc.s-separator* displaying the incidence relation of *sc-elements* is divided into:

- incidence sign of the “right” *sc-connector* – the incidence sign of the *sc-connector* whose *sc-identifier* is on the right, represented as “⊢”;
- incidence sign of the “left” *sc-connector* – the incidence sign of the *sc-identifier* whose *sc-identifier* is on the left, represented as “⊣”;
- incidence sign of the incoming *sc-arc* on the right is the incidence sign of the *sc-arc*, whose *sc-identifier* is on the right, represented as “| <”;
- incidence sign of the incoming *sc-arc* on the left is the incidence sign of the *sc-arc*, whose *sc-identifier* is on the left, represented as “> |”.

Specified *sc.s-separators* are similar to *sc.s-sentences* in terms of their syntactic structure, but in terms of their denotational semantics, unlike *sc.s-connectors*, they are not representations of corresponding *sc-connectors*.

In Figure 3, an image of *sc.s-connectors* of the Basic and Extended alphabet corresponding to *sc.g-connectors*, which are *sc.g-arc of membership*, is shown.

The equality sign is the *sc.s-separator* of two *sc-identifiers* which identify (name) the same entity and, therefore, are *sc-identifiers** (external unique images) of the same *sc-element*. Most often, one of these two sc-identifiers is a simple *sc-identifier* and the other is an *sc-expression*. Rarely, both of these *sc-identifiers* are *sc-expressions*. And quite rarely, they are both simple sc-identifiers. The latter indicates that both of these *sc-identifiers* are basic *sc-identifiers** of the same *sc-element*. An example:

SC-code = sc.s-text;;

Here, the first *sc-identifier* is a proper name and the second is a common noun.

When translating *sc.s-text* into the *SC-code*, the equality sign may at some stage be matched with an *sc-edge* which belongs to the synonymy* relation of the *sc-elements* identified by the *sc-identifiers* connected by the equality sign. However, in the next step, the specified *sc-edge* is removed, and the *sc-elements* connected by it are patched together. Thus, the *sc-edge* belonging to the synonymy* relation of sc-elements has not only denotational but also operational semantics.

An equality sign with inclusion is an image of an *sc-arc* belonging to an immersion* relation connecting two *sc-nodes* denoting *sc-texts*, the first of which is immersing and the second (in which specified *sc-arc* comes) is immersed, introduced into the first *sc-text*. The *sc-arc* belonging to the immersion* relation is interpreted as a command to immerse one sc-text into the composition of another. When this command is executed, (1) all sc-elements of the immersing *sc-text* become elements belonging to the immersing sc-text, (2) all synonymous *sc-elements* that happen to be part of the immersing *sc-text* are patched together, (3) the *sc-node* denoting the immersing *sc-text*, as well as the specification of this *sc-text* (including the list of all its *sc-elements*), is immersed in the history of the knowledge base evolution together with the specification of the event of immersion of the considered *sc-text* into the knowledge base.

In Figure 4, the *Alphabet of sc.s-connectors* corresponding to *sc.g-connectors* that are not *sc.arcs of membership* is shown.

The minimum semantically coherent fragment of *sc.s-text* is the *sc.s-sentence*; an *sc.s-sentence*, (1) consisting of either two *sc-identifiers* connected by an *sc.s-connector* or three *sc-identifiers* separated by *sc. separators* representing an incidence relation of *sc-elements* and (2) ending with a double semicolon.

It is easy to notice that simple *sc.s-sentences* are essentially the same as RDF triplets, except that a simple *sc.s-sentence* can be "unfolded" using *sc.s-sentence conversion** without changing its meaning, while an RDF-triplet cannot ensure that. This is one of the reasons why, unlike RDF triplets, in simple *sc.s-sentences*, *sc.s-connectors* and *sc.s-separators* displaying the sc-element incidence relation cannot be omitted, since they also show the direction of the relation they display between the *sc.s-elements*.

The operations defined on the set of *sc.s-sentences* can be divided into three groups:

- a group of conversion operations of *sc.s-sentences* consisting of a single operation;
- a group of combination operations of *sc.s-sentence*;
- a group of decomposition operations of *sc.s-sentences* and, in particular, decomposition operations of *sc.s-sentences*.

The list of operations defined on the set of *sc.s-sentences* is as follows:

- Conversion operation of the *sc.s-sentence**. Every *sc.s-sentence* (including the simple *sc.s-sentence*) can be transformed into a semantically equivalent *sc.s-sentence* by a conversion ('reversal') of the chain of *sc.s-sentence* components. Thus, for example, when converting ("unfolding") a simple *sc.s-sentence*
  - its first *sc-identifier* (the first component of this *sc.s-sentence*) becomes the third component of the converted *sc.s-sentence*;
  - its second *sc-identifier* (the third component of the original *sc.s-sentence*) becomes the first component of the "converted" one;
  - the second component of the original *sc.s-sentence* (*sc.s-connector* or sc.s-separator, representing the sc-element incidence relation connecting the above components) remains the second component of the converted *sc.s-sentence*, but it changes direction (" $\ni$ " is replaced by " $\in$ " and vice versa, " $\supset$ " by " $\subset$ " and vice versa, " $\Rightarrow$ " by " $\Leftarrow$ " and vice versa, etc.). We can talk not only about the conversion of *sc.s-sentence* but also about the conversion of *sc.s-connector*, the conversion of sc.s-separator displaying the incidence relation of sc.s-elements.
- The attachment operation of the *sc. s-sentence** is the operation of attaching two *sc.s-sentence* when the last component of the first sentence matches the first component of the second one*. As a result of performing this operation:
  - the first component of the second *sc.s-sentence* is deleted;
  - the rest of the second sentence is surrounded by the sc.s-delimiter of attached sentences " (* " and " *) ". The separator of *sc.s-sentences* " ;; " also

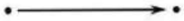
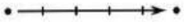
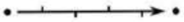
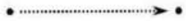
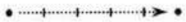
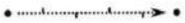
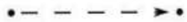
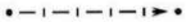
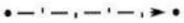
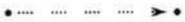
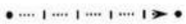
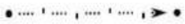
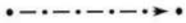
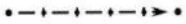
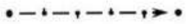
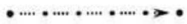
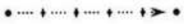
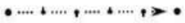
sc-element class	Image of *sc.g-connectors*	Image of *sc.n-connector* in Extended alphabet		Image of *sc.n-connector* in Base alphabet	
constant permanent positive sc-arc of membership		∋	∈	->	<-
constant permanent negative sc-arc of membership		∌	∉	-|>	<|-
constant permanent fuzzy sc-arc of membership		/∋	∈/	-/>	</-
constant temporal positive sc-arc of membership		~∋	∈~	~>	<~
constant temporal negative sc-arc of membership		~∌	∉~	~|>	<|~
constant temporal fuzzy sc-arc of membership		~/∋	∈/~	~/>	</~
variable permanent positive sc-arc of membership		_∋	_∈	_->	<-_
variable permanent negative sc-arc of membership		_∌	∉_	_-|>	<|-_
variable permanent fuzzy sc-arc of membership		_/∋	∈/_	_-/>	</-_
variable temporal positive sc-arc of membership		_~∋	∈~_	_~>	<~_
variable temporal negative sc-arc of membership		_~∌	∉~_	_~|>	<|~_
variable temporal fuzzy sc-arc of membership		_~/∋	∈/~_	_~/>	</~_
metavariable permanent positive sc-arc of membership		__∋	∈__	__->	<-__
metavariable permanent negative sc-arc of membership		__∌	∉__	__-|>	<|-__
metavariable permanent fuzzy sc-arc of membership		__/∋	∈/__	__-/>	</-__
metavariable temporal positive sc-arc of membership		__~∋	∈~__	__~>	<~__
metavariable temporal negative sc-arc of membership		__~∌	∉~__	__~|>	<|~__
metavariable temporal fuzzy sc-arc of membership		__~/∋	∈/~__	__~/>	</~__

Figure 3. An image of *sc.s-connectors* of the Basic and Extended alphabet corresponding to *sc.g-connectors*, which are *sc.g-arc of membership*

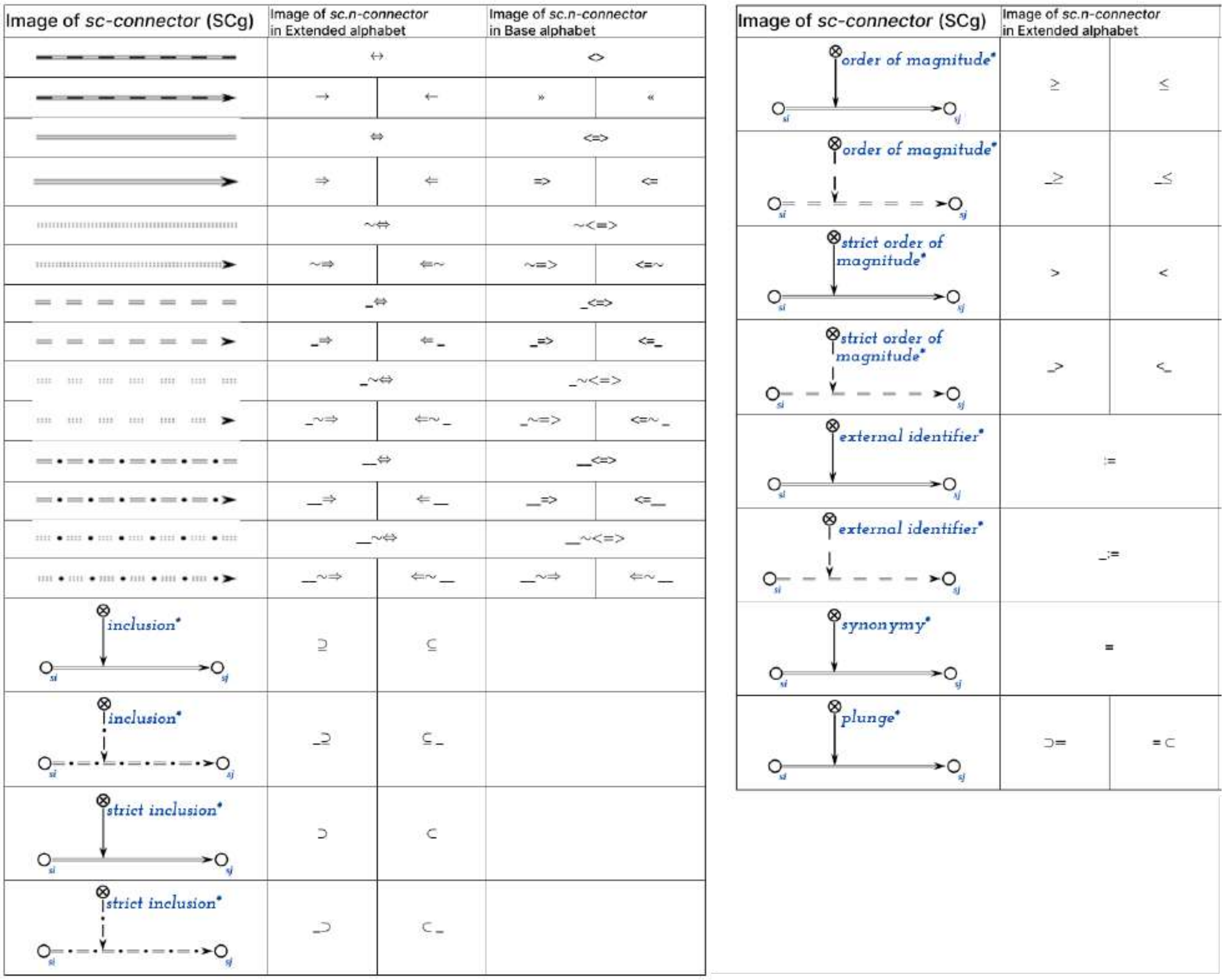


Image of *sc-connector* (SCg)	Image of *sc.n-connector* in Extended alphabet		Image of *sc.n-connector* in Base alphabet	
	↔		<>	
	→	←	»	«
	⇔		<=>	
	⇒	⇐	=>	<=
	~⇔		~<=>	
	~⇒	⇐~	~=>	<=~
	_⇔		_<=>	
	_⇒	⇐_	_=>	<=_
	_~⇔		_~<=>	
	_~⇒	⇐~_	_~=>	<=~_
	__⇔		__<=>	
	__⇒	⇐__	__=>	<=__
	__~⇔		__~<=>	
	__~⇒	⇐~__	__~⇒	⇐~__
inclusion* si sj	⊇	⊆		
inclusion* si sj	_⊇	⊆_		
strict inclusion* si sj	⊃	⊂		
strict inclusion* si sj	_⊃	⊂_		

Image of *sc-connector* (SCg)	Image of *sc.n-connector* in Extended alphabet	
order of magnitude* si sj	≥	≤
order of magnitude* si sj	_≥	_≤
strict order of magnitude* si sj	>	<
strict order of magnitude* si sj	_>	<_
external identifier* si sj	:=	
external identifier* si sj	_:=	
synonymy* si sj	=	
plunge* si sj	⊃=	=⊂

Figure 4. The Alphabet of *sc.s-connectors* corresponding to *sc.g-connectors* that are not *sc.arcs of membership*

falls inside the specified delimiter;
- the resulting construction is placed between the last component of the first sentence and the *sc.s-sentence* separator that ended the first sentence;
- the second sentence thus becomes an attached sc.s-sentence.

Similarly, any attached *sc. s-sentence* can be "docked" with other attached *sc.s-sentences*, in general, the level of such nesting is not limited.

- The merge operation of *sc.s-sentences** is the operation of attaching a simple *sc.s-sentence* to an sc.s-sentence where the last *sc.s-connector* is the same as the *sc.s-connector* of the simple *sc.s-sentence* and the sc-identifier preceding that *sc.s-connector* is the same as the first sc-identifier of the simple *sc.s-sentence*
  This operation causes the matching of *sc.s-identifiers* and *sc.s-connector* of the linked *sc.s-sentences* to be "patched" together, and the last *sc.s-identifiers* of the linked *sc.s-sentence* become the last components of the merged *sc.s-sentence*, separated by semicolons. In the same way, any number of simple *sc.s-sentence* can be attached.
- The decomposition operation* of *sc.s-sentences* into simple *sc.s-sentences*
  Every sc.s-sentence can be decomposed into a set of simple *sc.s-sentences*, i.e. represented as a sequence of simple *sc.s-sentences*.
- The decomposition operation of *sc.s-sentences* into simple *sc.s-sentences* with the sc.s-separator representing the incidence relation of the sc-elements*
  Each *sc.s-sentence* (including a simple sc.s-sentence with an *sc.s-connector*) can be represented as a semantically equivalent sequence of simple *sc.s-sentences with sc.s-separator* displaying the incidence relation of sc-elements. This operation uniquely generates a set of *simple sc.s-sentences* of the specified kind.

Obviously, the combination operations of *sc.s-sentences* and the decomposition operations of *sc.s-sentences* are inverse operations to each other.

From the semantic point of view, the sc.s-sentence is a description of some route in the corresponding sc-text, which is a graph structure of a special kind and whose structure is described (displayed) with *sc.s-sentences*. The specified route is “traversed” by sc-connectors and sc-element incidence relations, if the route passes through incident sc-connectors. The description of the specified route may additionally specify the sets (most often relations) to which the sc-connectors included in the described route belong. In addition, the specified route may have branches at the beginning and/or at the end, where any sc-element is equally incidental to several sc-connectors of the same type, connecting the specified sc-element to some other sc-elements. Thus, each specified branching consists of an unlimited number of branches, each of which consists of one sc-connector and one sc-element connected by it.

A sequence of *sc.s-sentences* separated by double dots forms the sc.s-text. Accordingly, the sc.s-sentence is the minimum sc.s-text.

The meaning of the sc.s-text (as well as the sc.s-text included in the structure) does not depend on the order of *sc.s-sentences* in these sc-texts. That is, rearranging *sc.s-sentences* within such sc.s-texts does not change the meaning of these sc.s-texts (i.e. leads to semantically equivalent sc.s-texts), but greatly affects the human perception (the “readability”) of these texts.

Similar to SCg-code, the SCs-code has a sublanguage – the SCs-code Core, which uses a minimal set of syntactic tools but has a semantic power equivalent to the power of SCs-code as a whole.

In the *SCs-code Core*:

- only simple *sc-identifiers* are used, including *sc-identifiers* of external ostis-files (sc-expressions are not used);
- only sc.s-separators are used, displaying the incidence relation of sc-elements, and *sc.s-connectors* displaying a constant permanent positive pair of membership (“ $\in$ ” and “ $\ni$ ” in the Extended Alphabet and “ $\dashrightarrow$ *or* ” and “ $\dashleftarrow$ ” in the Basic Alphabet). Other *sc.s-connector* are not used;
- sc.s-modifiers and, consequently, colons, which are a sign of completion of sc.s-modifiers, are not used;
- only simple *sc.s-sentences*, which, as follows from the above properties of the SCs-code Core, either consist of two simple sc-identifiers connected by a *sc.s-connector* representing a constant permanent positive pair of membership or three simple sc-identifiers separated by sc.s-separators representing an incendence relation of sc-elements are used.

It follows from the above properties of the *SCs-code Core* that in order to represent any sc-text by means of the *SCs-code Core* it is necessary for all sc-elements of this sc-text (except constant permanent positive pairs of membership) to build simple *sc-identifiers* corresponding to them, i.e. it is necessary to name all the specified sc-elements. In turn, the type of each used sc-element (except the constant permanent positive pairs of membership) is specified explicitly by indicating the membership of these elements to the corresponding sc-element classes, including the classes included in the *SC-code Core*.

As it is possible to notice from the above description, the *SCs-code Core* corresponds to the *SCg-code Core*, except that the *SCg-code Core* does not need to name all represented sc-elements, and also in the *SCg-code*, there are graphic images for sc-elements that belong to the corresponding classes of the *SC-code Core*, and this membership need not to be explicitly specified.

Obviously, it is inconvenient and inefficient to use the *SCs-code Core* for writing large fragments of knowledge bases in practice. Nevertheless, from a practical point of view, the *SCs-code Core* can be used, for example, to exchange information with third-party graph representation tools designed to represent information in the form of triplets (e.g., RDF storages). Syntactic extensions to the *SCs-code Core* are needed to enable wider practical usage with next purposes:

- to minimize the number of identifiable (named) sc-elements by using sc-expressions and eliminating the need to identify (name) all sc-elements;
- reducing text by minimizing the number of repetitions of the same sc-identifier by linking *sc.s-sentences*;
- to increase the visibility, “readability” of the sc.s-texts.

Next, let us consider the structured knowledge representation language of *ostis-systems* – an *SCn-code*.

VIII. SCn-code. Alphabet of the SCn-code and denotational semantics

An *SCn-code* is a language for the structured external representation of the *SC-code* texts, which is a syntactic extension of the *SCs-code*, aimed at increasing the clarity and compactness of the *SCs-code* texts.

The *SCn-code* allows switching from linear texts of the *SCs-code* to formatted and actually two-dimensional texts in which there appears a decomposition of the original linear text of the *SCs-code* into lines placed “vertically”. In this case, the beginning of all lines of text is fixed and defined by a known and limited set of rules, which makes it possible to use this when formatting sc.n-text (text belonging to the *SCn-code*).

An *SCn-code* is a language of two-dimensional texts. Accordingly, each text of such a language is defined by:

- a set of characters included in it;
- a “horizontal” character order (sequence) relation;
- a “vertical” character order (sequence) relation.

A character that is part of a two-dimensional text can generally have four “adjacent” characters:

- a character to its left within the same line;
- a character to its right within the same line;
- a character located strictly above it in the previous line;
- a character located strictly below it in the next line of text.

Due to the fact that sc.n-texts can include both sc.s-texts and sc.g-texts (delimited by the sc.n-contour), the *SCn-code* can be considered an integrator of different external knowledge representation languages. This makes it possible to compensate the disadvantages of one of the proposed options for external representation of sc-texts (*SCg-code*, *SCs-code*, *SCn-code*) with the advantages of other options when visualizing and developing the knowledge base of the *ostis-system*.

In this case, there is a transition from linearity of sc.s-texts to two-dimensionality of sc.n-texts.

An important feature of the *SCn-code* is the "two-dimensional" nature of its texts. This is manifested in that for each *SCn-code* fragment of text, the value of the indentation from the left edge of the line is essential.

In the *SCn-code* text, unlike the *SCs-code* text, the important thing for each text fragment is not only how this fragment is connected to other fragments "horizontally" (which fragment is to the left or to the right of the same line) but also how it is related to other fragments "vertically" (which fragment is higher on the previous line and which is lower on the next line), which fragment is below on the next line).

So, for example, if in the text of the *SCn-code* some sc-identifier(external sc-element identifier) is placed immediately after the vertical tab line and a certain *sc.s-connector* is placed exactly below it, it means that the specified sc-element is incident to the sc-connector represented by the specified *sc.s-connector*.

In order to provide the exact setting (formulation) of the rules of two-dimensional incidence elements (elementary fragments) of sc.n-texts, the concept of sc.n-text page is introduced, the concept of a line of sc.n-text, and also a special markup is used, which is vertical tab lines, the distance between which is approximately equal to the maximum length of the *sc.s-connector* (usually this distance equals the width of 4-5 characters).

The sc.n-text (text of the *SCn-code*) is a sequence of sentences of the *SCn-code*, each of which is not part of any other sentence in the sequence.

If the sc.n-text is part of some other file that is paginated, such as the publication of some part of a knowledge base, then the sc.n-page is only the part of the page that shows the sc.n-text, while the page of the specified file may be larger due to, for example, white fields on the edges of the page needed for subsequent printing.

The maximum number of characters in sc.n-text lines for each sc.n-text is fixed and is determined by the specific sc.n-text placement option. At the same time, depending on the indentation within a particular sc.n-text sentence, a line of sc.n-text may not start from the left edge of the sc.n-text (but always from some of the vertical markup lines) and have an arbitrary length limited by the right border of the sc.n-page.

A markup line is used to make sc.n-texts easier to read. The 1st markup line borders the left edge of the sc.n-page, the 2nd markup line is located approximately between the 5th and 6th characters of the line, and so on. The distance between the markup lines may vary depending on the font size but always remains the same within a single sc.n-text. The total number of markup lines is limited by the maximum possible width of the sc.n-page in the particular *ostis-system* file containing that sc.n-text.

The Alphabet of the *SCn-code* is the same as the Alphabet of the *SCs-code*. All components of sc.s-texts are also used in sc.n-texts:

- *sc-identifiers*;
- *sc.s-identifiers*;
- modifiers of *sc.s-connectors* with the corresponding delimiters (colons);
- separators used in sc-expressions denoting sc-multiples given by enumeration of elements with corresponding separators (semicolon or round marker);
- round markers in enumerations of sc-element identifiers linked by the same-type sc-connectors with the same-type modifiers to a given sc-element;
- sentence separators (double semicolons) (omitted when converting *sc.s-sentences* to sc.n-sentences);
- delimiters of attached *sc.s-sentences* (omitted when converting *sc.s-sentences* to sc.n-sentences).

However, unlike sc.s-texts in sc.n-texts:

- new kinds of sc-expressions (namely, sc-expressions that have a two-dimensional character) are added;
- a new kind of sentence separators – a blank line – is added;
- the placement of sentences, taking into account the two-dimensional nature of this placement, is changed.

New types of sc-expressions are added to the *SCn-code* compared to the *SCs-code*:

- an sc-expression, which is a two-dimensional *sc.n-text* delimited by an *sc.n-contour* or an *sc.n-frame*. Each *sc.n-contour* is represented conventionally as an opening curly bracket and a closing curly bracket located strictly below it through several lines. Inside these brackets (starting from the vertical markup line where the brackets are located to the right page edge), sc.n-text is placed. The resulting *sc.n-frame* is an image of the structure resulting from the translation of the specified *sc.n-text* into the *SC-code*. Each sc.n-frame is represented in the same way, only instead of curly braces it uses square brackets or square brackets with an exclamation mark (in the case of a sample file);

- an sc-expression, which is a two-dimensional sc.g-text delimited by an sc.n-contour or an sc.n-frame;
- an sc-expression, which is a two-dimensional graphical representation of an information construct encoded in some *ostis-system* file, delimited by the sc.n-frame. Such an information construction can be a table, a picture, a photograph, a diagram, a graph, and more.

It is easy to notice that an *sc.n-contour* is essentially the two-dimensional equivalent of the sc-expression structure, and an sc.n-frame is the two-dimensional equivalent of the *sc-expression of the inner file* of the *ostis-system* or *sc-expression* denoting the pattern file of the *ostis-system*.

From a formal point of view, an *sc.n-frame* is always a single line of *sc.n-text*. This means that the *sc.n-frame* cannot be syntactically divided into parts within the *sc.n-text* in which it is used and cannot be inserted inside it, for example, with attached *sc.n-sentences* or any other text (unless the *sc.n-frame* contains *sc.n-text*, but in this case specified *sc.n-text* will still be considered as a complete external file and not as a fragment of the surrounding *sc.n-text*).

The *sc.n-sentences* uses a delimiter that is a representation of the structure, which is called an *sc.n-contour*.

The concept of the *sc.n-sentence* is a natural generalization of the concept of the *sc.s-sentence*. Moreover, similarly for *sc.s-sentences*, the concept of concepts are introduced:

- of a simple *sc.n-sentence*;
- of a complex *sc.n-sentence*;
- of an *sc.n-sentence* containing attached *sc.n-sentence*;
- of an *sc.n-sentence* that does not contain any attached *sc.n-sentence*;
- of an attached *sc.n-sentence*;
- of unattached *sc.n-sentence*.

If each unattached *sc.s-sentence* is either the first sentence of the *sc.s-text* or begins after the *sc.s-sentence* separator (double semicolon), then each unattached *sc.n-sentence* starts at the beginning of a new line.

If each attached *sc.s-sentence* starts either after the opening delimiter (opening bracket with an asterisk) or after the separator of the
textitsc.s-sentence, then each attached *sc.n-sentence* starts on a new line under the *sc-identifier* that ends that *sc.n-sentence* (and accordingly, *sc.s-sentence*, respectively) in which this attached *sc.n-sentence* is embedded.

The first *sc-identifier* that is part of the *sc.n-sentence* before the *sc.s-connector* is highlighted in bold italics.

In *sc.n-sentences*, the double semicolon is not used as a sign of completion of these sentences and therefore is not used as a separator for *sc.n-sentences*. Such a separator is an empty line.

The two-dimensionality of the *SCn-code* gives more possibilities (degrees of freedom) for a clear and compact layout of the *sc.n-sentences*.

When the *sc.n-sentence* is drawn up, all the *sc.n-sentences* attached to it are clearly tabulated and attached to the original "vertical" one. The vertical tabulation line specifies the left border of the original (maximum) *sc.n-sentence* or the left border of the *sc.n-sentence* attached vertically.

The left border of the *sc.n-sentence* specifies the start of the first *sc.n-sentence* that is part of this *sc.n-sentence* and the start of the *sc.s-connector* that is incident to the specified *sc.s-identifier* and is placed strictly below this *sc-identifier*. The distance between the vertical tab lines is fixed and approximately equal to the maximum length of the *sc.s-connector*.

In contrast to *sc.s-texts*, in *sc.n-texts*, an *sc.s-connector* can be incident to the preceding *sc-identifier* (either simple one or an *sc-expression*) not only "horizontally" but also "vertically". To do this, the *sc.s-connector* is placed strictly below the *sc-identifier* that precedes it.

Also "vertical" *sc-identifier* can be incident to not one but several *sc.s-connectors*, which are consecutively "vertically" placed under the specified *sc-identifier*. This allows within one *sc.n-sentence* representing an arbitrary number of "branches" from each *sc-identifier*, i.e. an arbitrary number of *sc.s-connectors* incident to that *sc-identifier*.

Each *sc-identifier*, including the *sc-expression* delimited by curly or square brackets, must be placed immediately to the right of the vertical marking line if an *sc.s-connector* is placed below it.

Each *sc.s-connector* is highlighted in bold, non-cursive font and, if it is below an incident *sc-identifier*, is placed strictly between the two vertical marking lines, nestled to the left of these two marking lines.

Since in relation to the *SCn-code*, the *SCs-code* is the syntactic core of the language*, the *SCn-code* can be considered as the result of integrating several extensions of the *SCs-code* based on the syntactic transformation rules of *sc.s-texts* and *sc.n-texts*, oriented towards making better usage of those possibilities of visibility and compactness of *sc.n-texts* which are opened by the transition from linearity of *sc.s-texts* to two-dimensional *sc.g-texts*.

The list of operations defined on the set of *sc.n-sentences* is as follows:

- Transformation operation of *sc.s-sentence* to the *sc.n-sentence**
  Every *sc.s-sentence* written linearly ("horizontally") can be transformed into the corresponding two-dimensional *sc.n-sentence*. Let us list the basic rules for transforming *sc.s-sentences* into *sc.n-sentences*
  - The *sc.s-connector* can be placed on the next line below the preceding *sc-identifier*, starting from

the same character of the next line as the specified *sc-identifier*;

- If the *sc-identifier* is moved to the next line, it is continued on the next line with the same indentation from the beginning of the line as the specified *sc-identifier* starts;
- A semicolon-delimited listing of *sc-identifiers* can be carried out not "line by line" but "column by column" by placing each following *sc-identifier* strictly below the preceding one. In this case, the semicolon separator can be replaced by a circle marker placed in front of each *sc-identifier* to be enumerated;
- a closing curly or square bracket may be placed strictly below the corresponding opening bracket;
- The *sc-identifier* in the *sc.n-sentence* can be connected to other *sc-identifiers* via several different *sc.s-connectors*. In this case, each of these *sc.s-connector* is placed strictly below the preceding one but only after the recording of the entire, generally branched, chain of *sc.s-connector* and *sc-identifier* that starts with the preceding *sc.s-connector* is completed. In the *SCs-code*, there is no analogue to such sentences with the unrestricted possibility of describing "branched" connections of *sc-identifiers*. Consequently, if in *sc.s-text*, the *sc-identifier* can be incident to no more than two *sc.s-connectors* (to its left and right), then in *sc.n-text*, sc-identifier can additionally be incident to an unlimited number (not necessarily identical) of *sc.s-connectors* that are placed "vertically" strictly below it.

- Attachment operation of the *sc.n-sentence**
Some *sc.n-sentence* can be attached to another *sc.n-sentence* if this other *sc.n-sentence* has an sc-identifier (but not an sc.s-modifier) that begins the first (attachable) *sc.n-sentence*. Joining in is done as follows:
- The initial *sc-identifier* of the attached sentence is omitted;
- The remainder of the *sc.n-sentence*, starting from the *sc.s-connector*, is written under the same *sc.s-identifier* but forming part of the *sc.n-sentence* to which this *sc.n-sentence* is attached. All indents in the attached *sc.n-sentence* are shifted accordingly.
An arbitrary number of any number of branches can be formed in this way.

In essence, the semantics of the *sc.n-sentence* is the set of routes in *sc-text*, possibly intersecting and originating from a given *sc-element*.

IX. Example of the text represented in the SCg-code, SCs-code, and SCn-code

Let us consider the fragment of the *sc.g-text* shown in Figure 5. This fragment represents a class of material objects including: Earth, Moon, Sun, Mars. The material object "Moon" has two main identifiers, in Russian and English. "Earth" and "Mars" are related to "Sun" by a "revolve around*" relation. "Moon" is related to "Earth" using the "satellite*" relation.

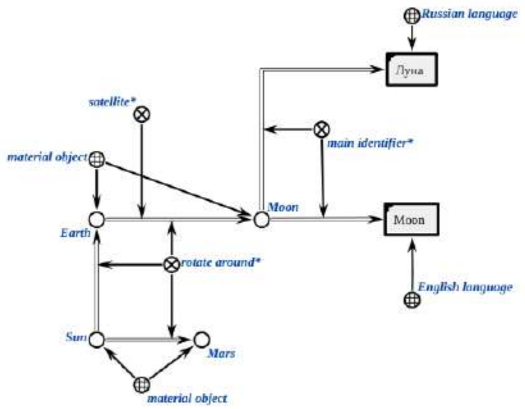
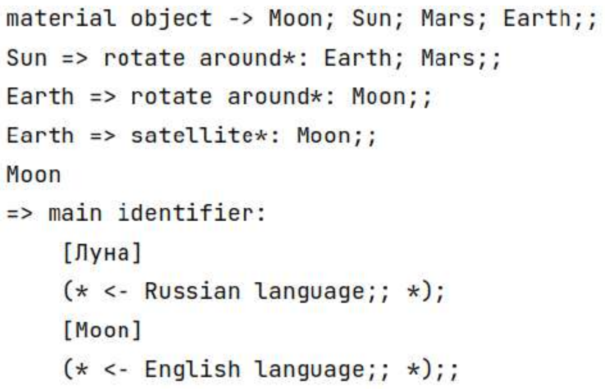


Figure 5. A fragment of the sc.g-text



```
material object -> Moon; Sun; Mars; Earth;;
Sun => rotate around*: Earth; Mars;;
Earth => rotate around*: Moon;;
Earth => satellite*: Moon;;
Moon
=> main identifier:
    [Луна]
    (* <- Russian language;; *);
    [Moon]
    (* <- English language;; *);;
```

Figure 6. A fragment of the sc.s-text

Any *sc.g-text* can easily be represented by the *sc.s-text*. Accordingly, the fragment of *sc.g-text* described above is represented in *sc.s-text* in Figure 6:

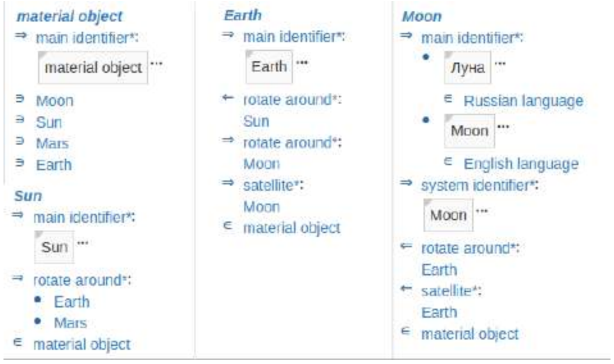
In Figure 7, a fragment of the above text in the *SCn code* is shown.

X. Conclusion

In this article, the concepts of internal and external languages of a next-generation intelligent computer system, the family of external languages of *ostis-systems* are considered. The syntax and denotational semantics of the *SCg-code*, *SCs-code*, *SCn-code* are clarified.

Examples of information constructions described with the *SCg-code*, *SCs-code*, *SCn-code* are given.

The results obtained will improve the future development of next-generation intelligent computer systems, as well as the compatibility and interoperability of the components of such systems.



material object
⇒ main identifier*:
material object ...
∋ Moon
∋ Sun
∋ Mars
∋ Earth

Sun
⇒ main identifier*:
Sun ...
⇒ rotate around*:
- Earth
- Mars

∈ material object

Earth
⇒ main identifier*:
Earth ...
← rotate around*:
Sun
⇒ rotate around*:
Moon
⇒ satellite*:
Moon
∈ material object

Moon
⇒ main identifier*:
- Луна ...
∈ Russian language
- Moon ...
∈ English language

⇒ system identifier*:
Moon ...
⇐ rotate around*:
Earth
← satellite*:
Earth
∈ material object

Figure 7. A fragment of the sc.n-text

References

[1] V. Martynov, *Universal Semantic Code (Grammar. Dictionary. Texts)*. Minsk: Nauka i tekhnika [Science and technics], 1977.

[2] V. V. Golenkov, N. A. Gulyakina, D. V. Shunkevich, *Open technology for ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, G. V.V., Ed. Minsk: Bestprint, 2021.

[3] P. Warren, P. Mulholland, T. Collins, and E. Motta, "Improving comprehension of knowledge representation languages: a case study with description logics," *International Journal of Human-Computer Studies*, vol. 122, 09 2018.

[4] P. P.-S. Chen, "The entity-relationship model—toward a unified view of data," *ACM Trans. Database Syst.*, vol. 1, no. 1, p. 9–36, mar 1976. [Online]. Available: https://doi.org/10.1145/320434.320440

[5] J. Sowa, *Conceptual Structures: Information Processing in Mind and Machine The Systems Programming Series*, 01 1984.

[6] J. E. Rumbaugh, M. R. Blaha, W. J. Premerlani, F. Eddy, and W. E. Lorenson, "Object-oriented modelling and design," 1991.

[7] G. Schreiber, B. J. Wielinga, and J. Breuker, "Kads : a principled approach to knowledge-based system development," 1993.

[8] G. Booch, J. Rumbaugh, and I. Jacobson, "Unified modeling language user guide, the (2nd edition) (addison-wesley object technology series)," *J. Database Manag.*, vol. 10, 01 1999.

[9] D. Ausubel, "Educational psychology: A cognitive view," 01 1968.

[10] D. F. Dansereau, "The development of a learning strategies curriculum," 1978.

[11] D. H. Jonassen, K. L. Beissner, and M. Yacci, "Structural knowledge: Techniques for representing, conveying, and acquiring structural knowledge," 1993.

[12] D. F. Dansereau and C. D. Holley, "Development and evaluation of a text mapping strategy," *Advances in psychology*, vol. 8, pp. 536–554, 1982.

[13] J. M. Scandura, "Structural learning theory: Current status and new perspectives," *Instructional Science*, vol. 29, no. 4, pp. 311–336, Jul 2001. [Online]. Available: https://doi.org/10.1023/A:1011995825726

[14] G. Paquette, "Building graphical knowledge representation languages-from informal to interoperable executable models," 01 2006.

[15] A. Boriskin, M. Sadouski, D. Koronchik, I. Zhukau, and A. Khusainov, "Ontology-based design of intelligent systems user interface," *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, pp. 95–106, 2017.

[16] V. V. Golenkov, N. A. Gulyakina, D. V. Shunkevich, *Open technology for ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, G. V.V., Ed. Minsk: Bestprint, 2021.

Семейство внешних языков интеллектуальных компьютерных систем нового поколения, близких языку внутреннего смыслового представления знаний

Жмырко А.В.

В работе рассматриваются понятия внешних и внутренних языков интеллектуальных компьютерных систем нового поколения. Описываются внешние языки представления знаний в рамках *Технологии OSTIS*, а именно *SCg-код*, *SCs-код*, *SCn-код*. Для каждого из внешних языков уточнены и детально рассмотрены синтаксис и денотационная семантика.

Received 10.09.2022

Representation of formal ontologies of basic entity classes in intelligent computer systems

Stanislau Butrin
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: stas.butrin1331@gmail.com

***Abstract*—In the article, the ontological approach to the design of knowledge bases of next-generation intelligent computer systems is considered. The main subject domains and ontologies containing the description of basic entity classes are allocated. The connections and relations between the basic concept classes are described. The results obtained will improve the efficiency of designing knowledge bases of intelligent computer systems.**

***Keywords*—knowledge base, ontology, top-level ontology, ontological approach to designing knowledge bases, basic entity class, subject domain.**

I. Introduction

The number of application fields for various computer systems increases every year together with the number of intelligent problems that require automation. This leads to the need to improve intelligent computer systems and expand their functionality.

A key element of such systems is knowledge bases, which largely determine the level of their intelligence [1].

A knowledge base is a systematized totality of knowledge stored in the memory of an intelligent computer system and sufficient to ensure purposeful (appropriate, adequate) functioning (behavior) of this system both in its external and internal environment (in its own knowledge base) [2].

To ensure the joint usage of different types of knowledge included in the knowledge base, it is necessary to ensure their compatibility with the specified knowledge base, which includes semantic compatibility that implies an unambiguous and unified interpretation of the used concepts for all fragments of the knowledge base.

II. Analysis of existing approaches to solving the problem

Among the variety of means for knowledge representation, the most effective are ontologies [3]. The essence of this approach when designing the knowledge base is to consider the knowledge base as a hierarchical system of selected subject domains and their corresponding ontologies. However, ontologically, it is possible to specify knowledge in different ways. To solve this problem, top-level ontologies are designed.

Let us consider the currently available top-level ontologies [4], [5]:

- **The Standard Upper Ontology** (SUMO) [6]
  - The key concept in the SUMO ontology is "Entity", and this concept includes the "Physical" and the “Abstract” concepts. The first category includes everything that has a position in space and time, and the second category includes all the rest.
  - The ontology covers the following areas of knowledge: general kinds of processes and objects, abstractions (set theory, attributes, relations), numbers and units of measurement, temporal concepts, parts and a whole, agents and intentions.
- **Descriptive Ontology for Linguistic and Cognitive Engineering** (DOLCE) [7]
  - A DOLCE ontology is focused on capturing the ontological categories underlying natural language and human common sense.
  - The top-level concept in the ontology is “Concrete”, indicating that all instances of this and its subtypes are private.
- **Cyc’s upper ontology** (OpenCyc) [8], [9]
  - At the top of the collection hierarchy is a universal collection named “Something”, which, by definition, contains everything that exists within the domain being described.
  - An OpenCyc knowledge base contains information from various subject domains: Philosophy, Mathematics, Chemistry, Biology, Psychology, Linguistics.

Usage of modern top-level ontologies in the development of knowledge bases of intelligent computer systems involves the problems of ensuring their compatibility. Since the original purpose of creating top-level ontologies was to ensure the compatibility of subject domain ontologies and applied ontologies but not the intelligent systems themselves.

Such problems are:

- unconstrained interpretation of concepts, caused by the lack of their clear definition;

- the lack of a unified technology of designing knowledge bases on the basis of top-level ontologies;
- the lack of association of top-level ontologies with any technology, which does not allow using them as reusable components.

Therefore, there is a need to develop such a top-level ontology system, which could provide semantic compatibility between a large number of ontologies of different subject domains.

III. Proposed approach

In this article, to solve the problems mentioned earlier, we propose to use an OSTIS Technology. This technology is a set of models, tools, and methods for the development of next-generation intelligent computer systems.

The proposed models are based on the following basic principles of the OSTIS Technology:

- using unified semantic networks with a basic set-theoretic interpretation of their elements as a method of knowledge representation;
- orientation on the semantic representation of knowledge;
- unification of knowledge base models of intelligent systems;
- orientation not only on the specification of knowledge but also on the specification of the model of how this knowledge will be processed.

The *OSTIS* Technology is based on the usage of unified semantic networks with a basic set-theoretic interpretation of their elements as a method of knowledge representation. This way of knowledge representation is called an *SC-code*, and the semantic networks, represented in the *SC-code*, are called *sc-graphs* (*sc-texts*, or *texts of the SC-code*). The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, can be *sc-arcs* or *sc-edges* depending on their orientation). The *Alphabet of the SC-code* consists of five basic elements, on the basis of which SC-code constructions of any complexity are built, including the introduction of more particular kinds of sc-elements (e.g., new concepts). The memory storing SC-code constructions is called semantic memory, or *sc-memory*.

The technology also offers several universal options for visualizing *SC-code* constructions, such as *SCg-code* (graphical variant), *SCn-code* (nonlinear hypertext variant), *SCs-code* (linear string variant).

Within this work, fragments of structured texts in the SCn-code [10], which are simultaneously fragments of the original texts of the knowledge base, understandable both to a human and to a machine, will often be used. This allows making the text more structured and formalized, while maintaining its readability. The symbol ":=" in such texts indicates alternative (synonymous) names of the described entity, revealing in more detail certain of its features.

The OSTIS Technology uses subject domains to formalize knowledge, allowing allocating from the diversity of the World only a certain class of entities under study, focusing attention only on something specific.

The proposed approach implies the development of families of Subject domains and ontologies, which would contain descriptions of all the necessary basic classes of entities for the building of the knowledge base of an intelligent computer system.

Such Subject domains and ontologies include:

- Subject domain and ontology of sets;
- Subject domain and ontology of connectives and relations;
- Subject domain and ontology of parameters, quantities, and scales;
- Subject domain and ontology of numbers and number structures;
- Subject domain and ontology of structures;
- Subject domain and ontology of temporal entities;
- Subject domain and ontology of temporal entities of ostis-systems knowledge bases;
- Subject domain and ontology of semantic neighborhoods;
- Subject domain and ontology of subject domains;
- Subject domain and ontology of ontologies;
- Subject domain and ontology of logical formulas, propositions, and formal theories;
- Subject domain and ontology of external information constructions and files of ostis-systems;
- Global subject domain of actions and problems and its corresponding ontology of methods and technologies.

These subject domains are part of the Knowledge base Kernel, which should be part of every intelligent system. This Kernel guarantees the compatibility of intelligent computer systems due to the common conceptual apparatus. Depending on the specifics of particular systems different Knowledge base Kernels can be allocated, but the presence of the basic part, which includes the subject domains and ontologies mentioned above, should remain unchanged.

The following Subject domains and ontologies are considered as part of this article:

- Subject domain and ontology of sets;
- Subject domain and ontology of connectives and relations;
- Subject domain and ontology of numbers and number structures;
- Subject domain and ontology of parameters, quantities, and scales;
- Subject domain and ontology of structures;
- Subject domain and ontology of temporal entities;
- Subject domain and ontology of situations and events, describing the dynamics of ostis-systems knowledge bases.

To each *subject domain*, it is possible to assign:

- a family of ontologies of different kinds corresponding to it;
- a set of semantic neighborhoods describing the research objects of this subject domain.

A ***Subject domain*** is a ***structure***, which includes:

- the main research (description) objects – primary and secondary ones;
- various classes of research objects;
- various connectives whose components are the research objects (both primary and secondary ones), and possibly other such connectives, that is, the connectives (as well as the research objects) may have different structural levels;
- different classes of the above connectives (i.e., relations);
- different classes of objects that are neither research objects nor the above-mentioned connectives but are components of these connectives.

For the formal specification of the relevant subject domain, a type of knowledge such as an *ontology* is used.

ontology
:= [sc-ontology]
:= [semantic specification of any knowledge that has a fairly complex structure, any coherent fragment of a knowledge base: a subject domain, a method for solving complex problems of some class, a description of a certain type of activity, a description of the area of performing a certain set of actions, language of problem-solving methods, etc.]

The concepts and relations considered in the subject domains are discussed in more detail in the Standard of the OSTIS Technology [2].

IV. Subject domain and ontology of sets

Subject domain of sets
:= [Set theory subject domain]
:= [Subject domain of the set theory]
:= [Subject domain whose research objects are sets]
∈ *subject domain*
∋ *maximum class of research objects'*:
set

This subject domain describes:

- concepts: finite set, infinite set, countable set, uncountable set, set without multiples, multiset, multiplicity of belonging, class, class of primary sc-elements, class of sets, class of structures, class of classes, fuzzy set, clear set, set of primary entities, family of sets, non-reflexive set, reflexive set, set of primary entities and sets, formed set, unformed set, empty set, singleton, pair, pair of different elements, triple;
- relations: belonging, inclusion, Cartesian product, Boolean, example, strict inclusion, combination, subdivision, intersection, pair of intersecting sets, pairwise intersecting sets, intersecting sets, pair of non-intersecting sets, pairwise non-intersecting sets, non-intersecting sets, difference of sets, symmetric difference of sets, family of subsets, equality of sets.

V. Subject domain and ontology of connectives and relations

Subject domain and ontology of connectives and relations
∈ *subject domain*
∋ *maximum class of research objects'*:
relation

This subject domain describes:

- concepts: binary relation, sc-connector, non-atomic binary relation, non-binary relation, non-oriented relation, oriented relation, class of equal-powered connectives, class of connectives of different power, unary relation, binary relation, quasi-binary relation, ternary relation, non-binary relation, reflexive relation, antireflexive relation, partially reflexive relation, symmetric relation, antisymmetric relation, partially symmetric relation, transitive relation, antitransitive relation, partially transitive relation;
- relation: relation attribute, first domain, second domain, relation composition, factor set, correspondence, correspondence relation, departure domain, arrival domain, image, prototype, surjective correspondence, non-surjective correspondence, all-round definite correspondence, partially definite correspondence, unambiguous correspondence, inverse correspondence, reversible correspondence, ambiguous correspondence, injective correspondence, one-to-one correspondence, set of combinations, set of placements, set of permutations.

VI. Subject domain and ontology of numbers and number structures

Subject domain of numbers and number structures
∈ *subject domain*
∋ *maximum class of research objects'*:
number

This subject domain describes:

- concepts: natural number, whole number, rational number, irrational number, real number, complex number, negative number, positive number, arithmetic expression, arithmetic operation, Pi number, zero, unit, minor unit, number structure, number system, decimal number system, binary number system, hexadecimal number system, fraction, regular fraction, decimal fraction, digit, Arabic digit, Roman digit;

- relations: opposite numbers, modulus, sum, product, exponentiation, greater than, equal to, greater than or equal to.

VII. Subject domain of parameters, values, and scales

This subject domain allows describing the properties and characteristics of objects, both qualitative and quantitative ones. The maximum class of research objects for the subject domain of parameters, values, and scales is the parameter.

Subject domain of parameters, values, and scales
:= [Subject domain of parameters and equivalence classes that are their elements (values, quantities)]
∈ *subject domain*
∋ *maximum class of research objects'*:
parameter

parameter
:= [characteristic]
:= [feature]
:= [property]
⇒ *explanation**:
[Each ***parameter*** is a class that is a family of all possible equivalence or tolerance classes defined by either a *equivalence relation* or a *tolerance relation* (symmetric, reflexive but partially transitive).]

Each particular parameter (characteristic), i.e. each element of a class of all possible parameters (characteristics) is essentially a sign of classifying entities with that characteristic according to the equivalence (similarity of value) of that characteristic. For example, the *color* parameter divides the set of entities with color into classes, each of which includes entities with the same color.

A description of the parameter example is shown in Figure 1.

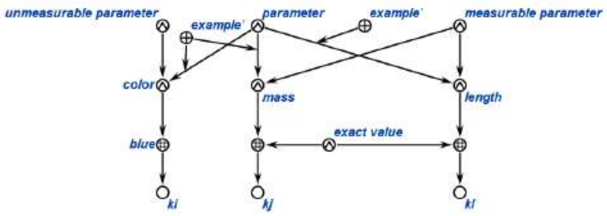


Figure 1. A description of the parameter example

value
:= [value of quantitative parameter]
⇒ *inclusion**:
- *exact value*
- *non-exact value*
- *interval value*

⇒ *explanation**:
[Each ***value*** represents an unambiguous and scale-independent measurement result for some characteristic of some entity]

exact value
:= [exact parameter value]
:= [set of all exact parameter values]
:= [the parameter value that is a family of equivalence classes corresponding to some equivalence relation]
:= [equivalence class]
⇒ *explanation**:
[Each ***exact value*** has one fixed value in some unit of measurement or on some scale. It is assumed that all elements of such a class have the same value of a given parameter, and deviations can be ignored.]

scale measurement
:= [scale]
⇐ *family of subsets**:
measurement
⇒ *explanation**:
[Each ***scale measurement*** is a subset of the *measurement* relation and is characterized not by a unit of measurement but by some reference point for that ***scale***. As the result of a ***scale measurement***, some point on the scale will serve, which is a certain distance away from the reference point in the required direction (smaller or larger).]

A description of an example of a scale measurement is shown in Figure 2.

In this example, ***ki*** denotes the class of entities that have a exact temperature of 330 K and ***bi*** is a specific example of such an entity.

This subject domain describes:

- concepts: measurable parameter, unmeasurable parameter, equivalence class level, non-exact value, interval value, parametric model, fixed unit measurement, arithmetic expression on values, arithmetic operation on values, action. measurement, problem. measurement;
- relations: definition area of a parameter, standard, measurement, exactitude, unit of measurement, zero mark, unit mark, sum of quantities, product of quantities, exponentiation of quantities, greater quantity, equality of quantities, greater or equal quantity.

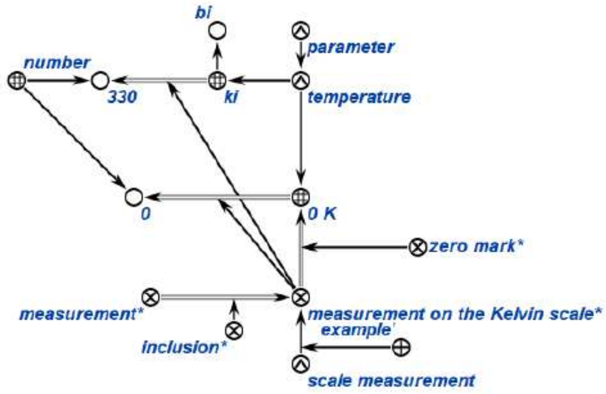


Figure 2. A description for an example of a scale measurement

VIII. Subject domain and ontology of temporal entities

The subject domain is necessary to describe entities and processes that occur in time, since the existing top-level ontologies do not handle this problem well and rarely operate anything other than the present tense.

Subject domain of temporal entities

:= [Subject domain of temporal connections and relations]

:= [Subject domain of temporal entities]

∈ *subject domain*

∋ *maximum class of research objects'*: *temporal entity*

temporal entity

:= [temporarily existing entity]

:= [non-stationary entity]

:= [entity that has either a beginning and/or an ending of its existence]

:= [sc-element that is a sign of some temporarily existing entity]

:= [entity with temporal characteristics (duration, starting point, ending point, etc.)]

⇒ *subdividing**:

{• *past entity*

• *present entity*

• *future entity*

}

⇒ *subdividing**:

{• *temporal relation*

• *temporal structure*

:= [structure containing at least one temporal entity]

⇒ *inclusion**: *structure*

⇒ *subdividing**:

{• *situation*

• *process*

}

• *material entity*

}

⇒ *subdividing**:

{• *continuous temporal entity*

⇒ *subdividing**:

{• *point temporal entity*

• *longtime continuous temporal entity*

}

• *discrete temporal entity*

:= [temporal entity that can be decomposed into a sequence of point temporal entities]

• *interrupted temporal entity*

:= [temporal entity with interrupts]

}

It should be noted that the above classification of *temporal entities* characterizes not so much the *temporal entities* themselves as our knowledge about them and the degree of granularity of knowledge about these entities with which they are described in the knowledge base. Thus, if it is not important for solving specific problems how a certain *temporal entity* changed within any period of time but only its initial and final state, then it can be considered as a *point temporal entity*.

This subject domain describes:

- concepts: past entity, present entity, future entity, process in sc-memory, process in the external environment of the ostis-system, material entity, influence, class of temporal relations, class of temporal and permanent relations, situational set, non-situational set, partial situational set, temporal connection, temporal relation, beginning, ending, duration, millennium, century, year, month, day, hour, minute, second;
- relations: influencing entity, influencing object, initial situation, causal situation, final situation, event, last added sc-element, temporal inclusion, temporal part, initial stage, final stage, intermediate stage, temporal inclusion without coincidence of initial and final moments.

IX. Subject domain of situations and events that describe the dynamics of ostis-systems knowledge bases

Since it is necessary to distinguish the temporal nature of the sc-element and the temporal nature of the entity denoted by this element, it becomes necessary to use the Subject domain of situations and events that describe the dynamics of ostis-systems knowledge bases to describe the dynamics of the knowledge base itself.

Subject domain of situations and events that describe the dynamics of ostis-systems knowledge bases

:= [Subject domain describing the dynamics of the knowledge base stored in sc-memory]
∈ *subject domain*
∋ *maximum class of research objects'*:
situation

elementary event in sc-memory
⊂ *event in sc-memory*
⇒ *explanation**:
[***elementary event in sc-memory*** is defined as an *event* that changes the state of only one *sc-element*]

.

⇒ *subdividing**:
{• *event of adding an sc-arc going out of a given sc-element*
• *event of adding an sc-arc coming into a given sc-element*
• *event of adding an sc-edge incident to a given sc-element*
• *event of deleting an sc-arc going out of a given sc-element*
• *event of deleting an sc-arc coming into a given sc-element*
• *event of deleting an sc-edge incident to a given sc-element*
• *event of deleting an sc-element*
• *event of changing the file contents*
}

This subject domain describes:

- concepts: sc-element, present sc-element, logically deleted sc-element, number, uncalculated number, calculated number, concept, main concept, non-main concept, concept going from main to non-main concept, concept going from non-main to main concept, specified entity, not enough specified entity, enough specified entity, medium specified entity, structure, file, event in sc-memory;
- relations: elementary event in sc-memory, event of adding an sc-arc going out of a given sc-element, event of adding an sc-arc coming into a given sc-element, event of adding an sc-edge incident to a given sc-element, event of deleting an sc-arc going out of a given sc-element, event of deleting an sc-arc coming into a given sc-element, event of deleting an sc-edge incident to a given sc-element, event of deleting an sc-element, event of changing the file contents.

X. Conclusion

In the article, an approach to designing top-level ontologies of knowledge bases of next-generation intelligent computer systems is considered. This approach is based on the representation of the knowledge base of intelligent computer systems grounded on the OSTIS Technology as a hierarchical structure of interrelated subject domains and their ontologies.

The main subject domains and ontologies containing the description of basic classes of entities are allocated. Connections and relations between basic concept classes are described.

These ontologies can be used to develop a universal Knowledge base Kernel, which will ensure interoperability of intelligent computer systems.

The results obtained will improve the efficiency of knowledge base development for next-generation intelligent computer systems, at the same time ensuring and preserving their compatibility.

The author would like to thank the research group of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for its help in the work and valuable comments.

References

[1] C. Gavrilova, *Gavrilova T.A. Knowledge Bases of Intelligent Systems / T.A. Gavrilova, V.F. Khoroshevsky. - SPb: Peter, 2000*, 2000.

[2] V. Golenkov, N. Guliakina, and D. Shunkevich, *Otkrytaja tehnologija ontologicheskogo proektirovanija, proizvodstva i jekspluatacii semanticheski sovmestimyh gibridnyh intellektual'nyh komp'juternyh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems]*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[3] I. Davydenko, "Ontology-based knowledge base design," *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, pp. 42–50, 2018.

[4] C. Partridge, *A survey of Top-Level Ontologies*. The Construction Innovation Hub, 2020.

[5] V. Mascardi, *A Comparison of Upper Ontologies*. DBLP, 2007.

[6] (2022, Nov) Suggested Upper Merged Ontology (SUMO). [Online]. Available: https://github.com/ontologyportal/sumo

[7] (2022, Nov) DOLCE: Descriptive Ontology for Linguistic and Cognitive Engineering. [Online]. Available: http://www.loa.istc.cnr.it/dolce/overview.html

[8] (2022, Nov) Cycorp – Cycorp Making Solutions Better. [Online]. Available: https://cyc.com/

[9] (2022, Nov) Github: Opencyc. [Online]. Available: https://github.com/asanchez75/opencyc

[10] (2021, Jun) IMS.ostis Metasystem. [Online]. Available: https://ims.ostis.net

Представление формальных онтологий базовых классов сущностей в интеллектуальных компьютерных системах

Бутрин С. В.

В работе рассмотрен онтологический подход к проектированию баз знаний интеллектуальных компьютерных систем нового поколения. Выделены основные предметные области и онтологии, содержащие описание базовых классов сущностей. Полученные результаты позволят повысить эффективность разработки баз знаний интеллектуальных компьютерных систем.

Received 01.11.2022

Structure of knowledge bases of next-generation intelligent computer systems: a hierarchical system of subject domains and their corresponding ontologies

Kseniya Bantsevich
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: ksusha.bantsevich@gmail.com

***Abstract*—The article is dedicated to the ontological approach to the design of knowledge bases of next-generation intelligent computer systems. This approach is based on the representation of the knowledge base as a hierarchical structure of interrelated subject domains and their ontologies built on the basis of top-level ontologies.**

***Keywords*—knowledge base, ontology, top-level ontology, ontological approach to knowledge base design, knowledge, structure, semantic neighborhood, subject domain.**

I. Introduction

The development of information technologies has led to the expansion of the variety of information used and, as a result, to the need to create intelligent systems capable of operating voluminous information resources. The most important types of such resources are knowledge bases.

The knowledge base is a systematized totality of knowledge stored in the memory of an intelligent computer system and sufficient to ensure the purposeful (appropriate, adequate) functioning (behavior) of this system both in its external and internal environment (in its own knowledge base).

An important stage in the development of knowledge bases of intelligent systems is their structuring. Structuring the database, i.e. the allocation of various interconnected substructures in it, is necessary for a number of reasons. In particular, this is necessary to ensure their syntactic compatibility, which implies the unification of the form of knowledge representation.

II. Analysis of existing approaches to solving the problem

To date, there are dozens of models of knowledge representation, each of which is adapted to represent a certain kind of knowledge, while when creating intelligent systems, it often becomes necessary to represent different types of knowledge within a single base. However, currently, none of the existing models, taken separately, can provide this.

In this regard, there is a need to create a universal structured model of knowledge representation, which would allow representing any kind of knowledge in a unified form.

Today, ontologies are the most effective means of structuring various areas of knowledge. The essence of the ontological approach when designing the knowledge base is to consider the structure of the knowledge base as a hierarchical system of allocated subject domains and their corresponding ontologies [1]. However, ontologically, there are many ways in which it is possible to describe the real world as it is. The solution to this problem is the usage of top-level ontologies [2] in the design of knowledge bases of intelligent computer systems.

A competently constructed top-level ontology will allow for broad syntactic compatibility between a large number of ontologies for various subject domains, since the terms of domain-oriented ontologies are subordinate to the terms of higher-level ontology.

At the moment, there are several developed top-level ontologies [3], [4]:

- **Descriptive Ontology for Linguistic and Cognitive Engineering** (DOLCE) [5]
  - The DOLCE ontology is focused on embracing the ontological categories underlying natural language and human common sense.
- **The Standard Upper Ontology** (SUMO) [6]
  - The SUMO ontology was created by combining publicly available ontological contents into a single, comprehensive, and coherent structure.
  - The ontology covers the following areas of knowledge: general types of processes and objects, abstractions (set theory, attributes, relations), numbers and units of measurement, temporal concepts, parts and a whole, agents and intentions.
- **Cyc's upper ontology** (OpenCyc) [7], [8]

- The key concept in the OpenCyc ontology is a collection, which can contain subcollections and instances, which, in turn, can act as any terms of the ontology.
- The OpenCyc knowledge base contains information from various subject domains: Philosophy, Mathematics, Chemistry, Biology, Psychology, Linguistics.

The list represented is not final.

There are more than fifteen top-level ontologies [4], the purpose for creation of which is to use them when creating lower-level ontologies. However, attempts to create a universal top-level ontology capable of ensuring the compatibility of intelligent computer systems have not led to the expected results, as they have a number of key disadvantages:

- Each of the represented ontologies is a monolithic structure in which there is no clear localization into separate small ontologies.
  The main problem in designing fragments of knowledge bases using the ontological approach is to allocate ontologies in such a way that they allow for the relatively independent evolution of each fragment. The data structure of top-level ontologies is a hierarchy consisting of a large number of different concepts. This type of structuring leads to a situation where the need to make changes in one place will necessarily entail the impossibility of editing another part of the ontology. Due to the above, this type of structuring makes ontologies inconvenient for their usage in the development of various intelligent systems.
- The top-level ontologies in question are not part of a complex technology.
  Since the named ontologies are not part of some complex technology, they cannot be considered as part of a library of reusable components, which, in turn, leads to inconveniences in the form of the need to adapt the ontologies used for each specific system.
- There are no knowledge base design technologies based on top-level ontology data.
  The lack of knowledge base design technologies makes it difficult to develop intelligent systems.

The lack of a satisfactory solution to these problems leads to incompatibility of the developed intelligent computer systems. Based on the above, there is a need to build such a system of top-level ontologies that could provide syntactic compatibility between a large number of ontologies of various subject domains in knowledge bases of intelligent computer systems.

III. Proposed approach

Within this work, it is proposed to take as a basis the approach developed within the *OSTIS Technology* [9] to the development of knowledge bases of next-generation intelligent computer systems. The proposed models are based on the following basic principles of the *OSTIS Technology*:

- the usage of an ontological approach to the design of knowledge bases, which involves structuring the knowledge base grounded on ontologies;
- focus on the possibility of collective design of knowledge bases within the project;
- orientation to the semantic representation of knowledge;
- unification of knowledge base models of intelligent systems.

To solve the above problems, it is necessary:

- to formally clarify and coordinate the interpretation of such concepts as *structure*, *semantic neighborhood*, *subject domain*, *ontology*, since these concepts are the basic classes of entities that form the basis for structuring knowledge bases of intelligent systems;
- to develop top-level ontological models for structuring the knowledge base grounded on the allocated concepts.

The ontological model built on the basis of these concepts will become the Kernel of the knowledge base, ensuring the compatibility of intelligent systems due to the unified representation of knowledge. It should be noted that, depending on the specifics of the systems being developed, their knowledge bases may expand, however, the ontological model underlying the *Kernel*, will ensure further compatibility of the systems being developed.

The approach proposed in this article is based on the ideas of building systems based on semantic networks implemented in the *OSTIS Technology*. This technology is a complex of models, tools, and methods designed for the development of intelligent computer systems, as well as for the constant updating and improvement of the technology itself.

The *OSTIS* Technology is based on the usage of unified semantic networks with a basic set-theoretic interpretation of their elements as a method of knowledge representation. This way of knowledge representation is called an *SC-code*, and the semantic networks, represented in the *SC-code*, are called *sc-graphs* (*sc-texts*, or *texts of the SC-code*). The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, can be *sc-arcs* or *sc-edges* depending on their orientation). The *Alphabet of the SC-code* consists of five basic elements, on the basis of which SC-code constructions of any complexity are built, including the introduction of more particular kinds of sc-elements (e.g., new concepts). The memory storing SC-code constructions is called semantic memory, or *sc-memory*.

The key feature of the *SC-code* is the joint usage of the mathematical apparatus of a graph theory and a set theory. This allows, on the one hand, ensuring the strictness and universatility of formalization tools and, on the other hand, ensuring the convenience of storing and processing information represented in this form.

Within the technology, several universal variants of visualization of the *SC-code* constructions are also proposed, such as *SCg-code* (graphic version), *SCn-code* (non-linear hypertextual version), *SCs-code* (linear string version).

The basis of the knowledge base within the *OSTIS Technology* is a hierarchical system of subject domains and ontologies.

Within this article, fragments of structured texts in the SCn-code [10] will often be used, which are simultaneously fragments of source texts of the knowledge base, which are understandable both to a human and to a machine. This allows making the text more structured and formalized while maintaining its readability. The symbol ":=" in such texts indicates alternative (synonymous) names of the described entity, which reveal in more detail some of its features.

Next, we will take a closer look at the fragments of sc-models of these top-level ontologies proposed within the *OSTIS Technology*.

IV. Concept of knowledge and formal models of ostis-systems knowledge bases

Within the model of the ostis-systems knowledge base, syntactically correct (for the corresponding language) and semantically integral information constructions are distinguished. We will call such constructions *knowledge*.

knowledge
⊂ *information construction*
⇒ *coverage**:
knowledge type
:= [Set of various knowledge types]

The fact that a family of knowledge types is a covering of a Set of various knowledge means that each knowledge belongs to at least one knowledge type that we have identified.

knowledge type
∋ *specification*
:= [description of the specified entity]
⊃ *specification of a material entity*
⊃ *specification of an inverse entity that is not a set*
⊃ *specification of a geometric point*
⊃ *specification of the number*
⊃ *specification of the set*
⊃ *connection specification*
⊃ *structure specification*
⊃ *class specification*
⊃ *specification of a class of entities that are not sets*
⊃ *relation specification*
⊃ *specification of the class of classes*
⊃ *specification of the class of structures*
⊃ *specification of concepts*
⊃ *semantic neighborhood*
⊃ *unique specification*
∋ *formal theory*
∋ *subject domain*
∋ *subject domain and ontology*
:= [subject domain and its ontology]
:= [subject domain and its corresponding unified ontology]
∋ *meta-knowledge*
:= [knowledge specification]
⊃ *ontology*
⊃ *ontology of the subject domain*
⊃ *structural ontology of the subject domain*
⊃ *set-theoretic ontology of the subject domain*
⊃ *logical ontology of the subject domain*
⊃ *terminological ontology of the subject domain*
⊃ *unified ontology of the subject domain*
∋ *problem*
∋ *plan*
∋ *protocol*
∋ *method*
∋ *technology*
∋ *knowledge base*

Even a small list of *knowledge types* indicates a huge variety of *knowledge types*.

Knowledge is divided into *declarative* and *procedural*. *Declarative knowledge* is understood as *knowledge* that has only *denotational semantics*, which is represented as a semantic *specification* of the concepts system used in this *knowledge*. *Procedural knowledge* is meant as *knowledge* that has not only *denotational semantics* but also *operational semantics*, which is represented as a family of *agents specifications* that interpret *procedural knowledge* aimed at solving some initiated problem.

Within the *OSTIS Technolog*y, relations defined on a set of knowledge are also distinguished.

relation defined on a set of knowledge
∋ *child knowledge**
:= [knowledge that inherits from the "maternal" knowledge all the properties of the research objects described there]
⊃ *child section**

⊃ *private subject domain and ontology**
∋ *specification**
:= [be knowledge, which is a specification (description) of a given entity]
∋ *ontology**
:= [be a semantic specification of a given knowledge*]
∋ *semantic equivalency**
∋ *therefore**
∋ *logical equivalency**

V. Concept of a structure

Existing approaches to the development of *knowledge bases* are grounded on considering specific elements of the knowledge base (classes, instances, relations, etc.) as specification objects. However, with the accumulation of large amounts of information in the *knowledge base*, it becomes necessary to allocate entire fragments of the *knowledge base* and be able to specify them, considering them as separate entities. The concept of a *structure* is the basis for the representation of *knowledge*, *meta-knowledge*, and their structuring.

The concept of a *structure* is one of the most general concepts (from the point of view of clarifying semantics) when describing the properties of an object.

By *structure* we mean a set of *sc-elements*, the removal of one of which may lead to a violation of the integrity of this set.

Let us consider the typology of the *structures* described in the *knowledge base*:

structure
⇒ *subdividing**:
{• *connected structure*
• *disconnected structure*
}

The structure represented in the SC-code will be matched with an orgraph whose vertices are sc-elements and arcs are connectives of incident relations connecting sc-connectors with incident sc-elements, which are components of these sc-connectors. If the orgraph obtained in this way is a connected orgraph, then the initial structure will be considered a *connected structure*. If the orgraph obtained in this way is not a connected orgraph, then the initial structure will be considered a *disconnected structure*.

structure
⇒ *subdividing**:
{• *trivial structure*
• *non-trivial structure*
}

A *trivial structure* is a *structure* that does not contain connectives as elements. In turn, a *non-trivial structure* means a *structure*, among the elements of which there is at least one connective.

On the basis of stationarity/nonstationarity, *dynamic structures* (processes) are distinguished – *structures* whose composition changes over time, as well as *static structures* – *structures* whose composition does not change over time.

structure
⇒ *subdividing**:
{• *process*
:= [dynamic structure]
• *static structure*
:= [stationary structure]
}

structure
⇒ *subdividing**:
{• *temporal structure*
• *permanent structure*
}

For the formal representation of *structures*, concepts describing elements within the structure were introduced:

structure element′
⇒ *subdividing**:
{• *unrepresented set′*
:= [set that is not represented within this structure′]
• *fully represented set′*
:= [set fully represented within this structure′]
• *partially represented set′*
:= [set partially represented within this structure′]
• *structure element that is not a set′*
}

A number of correspondences can be determined between structures, such as *homomorphism*, *polymorphism*, *automorphism*, *isomorphism*, as well as *similarity of structures*, which allows fixing the fact that there is some analogy, similarities, and differences of some substructures of the *structures* under consideration.

VI. Concept of a semantic neighborhood

For the specification of particular entities within the knowledge base, the concept of a *semantic neighborhood* is introduced. A *semantic neighborhood* is a specification (description) of a given entity, the sign of which is indicated as a key element of this specification.

The set of attributes by which entities can be specified is different. In addition, it may be necessary to specify the same entity in different aspects and explicitly record these aspects in the knowledge base.

A *semantic neighborhood* is a specification of a given entity, the sign of which is indicated as a key element of this specification. Unlike other knowledge types, semantic neighborhood has only one key element.

There are complete and basic semantic neighborhoods.

complete semantic neighborhood
:= [full specification of some described entity]

The structure of the *full semantic neighborhood* is determined primarily by the semantic typology of the entity being described. So, for example, for a concept, it is necessary to include the following information in the *full semantic neighborhood* (if available):

- identification options in various external languages (sc-identifiers);
- belonging to a certain subject domain with an indication of the role performed within this subject domain;
- set-theoretic connections of a given concept with other sc-elements;
- definition or explanation;
- propositions describing the properties of the specified concept;
- problems and their classes, in which this concept is a key element;
- description of a typical example for using this concept;
- instances of the described concept.

For a concept that is a relation, the following is additionally specified:

- domains;
- scope of definition;
- relation diagram;
- classes of relations to which the described relation belongs.

basic semantic neighborhood
:= [minimally sufficient semantic neighborhood]
:= [minimum specification of the described entity]

The structure of the *basic semantic neighborhood* is determined primarily by the semantic typology of the entity being described. For example, for a concept, the following information should be included in the *basic semantic neighborhood* (if available):

- identification options in various external languages (sc-identifiers);
- belonging to a certain subject domain with an indication of the role performed within this subject domain;
- definition or explanation.

For a concept that is a relation, the following is additionally specified:

- domains;
- scope of definition;
- description for a typical example of a connective of the specified relation (specification of a typical instance).

Also, a *specialized semantic neighborhood* is distinguished – a type of the *semantic neighborhood*, the set of relations for which is specified separately for each type of such a neighborhood.

specialized semantic neighborhood
⊃ *explanation*
⊃ *note*
⊃ *description of a typical instance*

The concept of a *semantic neighborhood*, supplemented by the clarification of such concepts as semantic distance between signs (semantic proximity of signs), the radius of the semantic neighborhood, is a promising basis for the study of the properties of semantic space.

VII. Concept of a subject domain

The most important stage in the development of knowledge bases is the process of identifying the *subject domains* described and their representation in the knowledge base.

The concept of the ***subject domain*** is the most important methodological technique that allows distinguishing only a certain class of entities under study and only a certain family of relations set on the specified class from the whole variety of the World, that is, localization is carried out, focusing attention only on this, abstracting from the rest of the studied World.

Each *subject domain* can be matched to:

- a family of corresponding ontologies of different types;
- a set of semantic neighborhoods describing the research objects in this subject domain.

A ***Subject domain*** is a ***structure***, which includes:

- the main research (description) objects – primary and secondary ones;
- various classes of research objects;
- various connectives whose components are the research objects (both primary and secondary ones), and possibly other such connectives, that is, the connectives (as well as the research objects) may have different structural levels;
- different classes of the above connectives (i.e., relations);
- different classes of objects that are neither research objects nor the above-mentioned connectives, but are components of these connectives.

At the same time, all classes declared by the concepts under study must be fully represented within this subject domain together with their elements, elements of elements, etc. up to terminal elements.

Each knowledge type can be matched with a subject domain, which is the result of integrating all knowledge of this type. This knowledge becomes the research object within the specified subject domain.

subject domain

:= [connections system of a certain set of research objects, the *key elements* of which are:

- classes (more precisely, class signs) of research objects (objects described by this subject domain);
- specific research objects with special properties;
- classes of connections that are part of the system under consideration – relations defined on the set of elements of the system under consideration;
- parameters specified on a set of elements of the system under consideration;
- classes of structures that are fragments of the system under consideration.

]

:= [structure representing a set of connections (more precisely, the signs of connections) and the corresponding set of components of these connections, which include:

- elements (instances) of some specified classes of research objects (primary entities under study);
- the connections themselves that are part of the specified structure;
- introduced classes of research objects;
- introduced relations (connection classes);
- introduced parameters (classes of equivalent entity classes);
- parameter values (and, in particular, values for the measured parameters);
- introduced structures that are fragments (substructures) of the structure under consideration;
- introduced classes of substructures of the structure under consideration.

]

The *subject domains* allocated within the *knowledge base* of the intelligent system and their corresponding *ontologies* are a kind of semantic strata, clusters that allow "decomposing" all *knowledge* stored in memory on "semantic shelves" in the presence of clear criteria that allow unambiguously determining on which "shelf" should this or that *knowledge* be placed.

According to the level of research attention, concepts within the subject domain can perform the following roles:

role of the subject domain element

:= [role relation that links subject domains with their key signs]

:= [role of the key element (the sign of the key entities) of the subject domain]

:= [role of the key sign of the subject domain]

∋ *class of research objects′*
 := [be a class of primary (for a given subject domain) research objects′]

∋ *maximum class of research objects′*
 := [class of research objects for which in the specified (!) subject domain there is no other class of research objects that would be its superset′]

∋ *key research object′*
 := [special research object′]
 := [be a sign of a special research object within a given subject domain′]
 := [research object with special properties′]

∋ *concept used in the subject domain′*
 := [concept used in a given subject domain not as one of the research objects but as a key concept′]

∋ *primary research element of the subject domain′*
 := [sign of the primary research object within a given subject domain′]

∋ *secondary research element of the subject domain′*
 := [sign of the secondary research object within the subject domain′]

∋ *non-investigated element of the subject domain′*
 := [auxiliary element of a subject domain being investigated in another (adjacent) subject domain′]

The following types of subject domains are distinguished:

subject domain

⇒ *subdividing**:

{• *static subject domain*
  := [stationary subject domain]
  := [*subject domain*, in which the relations between the entities that are part of it do not depend on time (do not change in time); *temporal entities* cannot be the elements of the ***static subject domain***]

• *quasi-static subject domain*
  := [*subject domain*, the solution of problems in which does not require taking into account the temporal properties of research objects]

• *dynamic subject domain*

:= [non-stationary subject domain]
:= [*subject domain*, which describes a change in the state (including the internal structure) of research objects and/or a change in the configuration of connections between research objects]
:= [*subject domain*, in which some relations between entities that are part of it change over time (that is, they are situational, non-stationary in nature, in other words, they are *temporal entities*)]
}
⇒ *subdividing**:
{• *primary subject domain*
:= [*subject domain*, the research objects of which are external entities (denoted by primary *sc-elements*)]
• *secondary subject domain*
:= [meta-subject domain]
:= [*subject domain*, the research objects of which are *sc-sets* (relations, parameters, structures, classes of structures, knowledge, languages, etc.)]
}

In all the variety of subject domains, a special place is occupied by:

- the ***Subject domain of subject domains***, the research objects of which are all kinds of subject domains and the research subject are all kinds of role relations linking subject domains with their elements, relations linking subject domains with each other, relation linking subject domains with their ontologies;
- the ***Subject domain of entities***, which is the subject domain of the highest level and defines the basic semantic typology of sc-elements (signs included in the texts of the SC-code);
- a family of *subject domains*, each of which defines the semantics and syntax of some *sc-language* that provides a representation of *ontologies* of the appropriate type (for example, set-theoretic ontologies of terminological ontologies);
- a family of *top-level subject domains*, in which the classes of research objects are very "large" entity classes. Such classes, in particular, include:
  – class of various material entities,
  – class of various sets,
  – class of various connections,
  – class of various relations,
  – class of various structures,
  – class of various temporal (non-stationary) entities,
  – class of various actions (influences),
  – class of various parameters (characteristics),
  – class of various knowledge,
  – etc.

It is important to note that a *subject domain* can also be considered as a *semantic neighborhood* if we consider its center to be the sign of an entity that is the maximum class of research objects.

VIII. Concept of an ontology

For the formal specification of the corresponding subject domain, focused on the description of the properties and relations of the concepts that make up the specified subject domain, such a knowledge type as *ontology* is used.

Ontologies are the most important *knowledge type*, providing semantic systematization of *knowledge* stored in memory of *intelligent computer systems* (including *ostis-systems*) and, accordingly, semantic structuring of *knowledge bases*.

ontology
:= [sc-ontology]
:= [semantic specification of any knowledge having a sufficiently complex structure, of any integral fragment of the knowledge base — a subject domain, a method for solving complex problems of a certain class, a description for the history of a certain activity type, a description for the scope of a certain set of actions (problem-solving areas), a representation language for problem-solving methods, etc.]
:= [semantic *specification* of some enough informative resource (knowledge)]
⊂ *specification*
⊂ *meta-knowledge*
∈ *knowledge type*
:= [most important *meta-knowledge* type included in the knowledge base]
:= [specification (clarification) of the *concepts* that system used in the corresponding (specified) *knowledge*]

The ontology includes:

- the typology of the specified knowledge;
- connections of the specified knowledge with other knowledge;
- the specification of key concepts used in the specified knowledge, as well as key instances of some such concepts.

It is important to note that if a *specification* can specify (describe) any *entity*, then an *ontology* specifies only various *knowledge*. At the same time, the most important objects of such a specification are *subject domains*.

The main *purpose* of building *ontology* is semantic clarification (explanation, and ideally definition) of such a family of *signs* used in given *knowledge*, which are sufficient to understand the meaning of all specified *knowledge*. As it turns out, the number of characters whose meaning determines the meaning of all specified *knowledge* is not large.

ontology

⇒ *subdividing**:

{● *informal ontology*

● *formal ontology*

:= [ontology represented in a formal language]

:= [formal description of the denotational semantics (semantic interpretation) of the specified knowledge]

}

Obviously, in the absence of sufficiently complete formal ontologies, it is impossible to ensure semantic compatibility (integrability) of various knowledge stored in the knowledge base, as well as acquired from the outside.

An *ontology* is most often interpreted as a specification of conceptualization (specification of a *concepts* system) of a given *subject domain*. Here we mean a description of the set-theoretic relations (first of all, the classification) of the *concepts* used, as well as a description of various regularities for entities belonging to these *concepts*. However, important types of the *subject domain* specification are also:

- a description of the relations of the specified *subject domain* with other *subject domains*;
- a description of the terminology of the specified *subject domain*.

ontology of the subject domain

:= [description of the *denotational semantics* of the language being defined (set) by the corresponding (specified) *subject domain*]

:= [information suprastructure (meta-information) over the corresponding (specified) *subject domain*, describing various aspects of this *subject domain* as a sufficiently large, self-sufficient, and semantically integral fragment of the *knowledge base*]

:= [meta-information (meta-knowledge) about some *subject domain*]

The *ontology of the subject domain* can be interpreted, on the one hand, as a *semantic neighborhood* of the corresponding *subject domain* and, on the other hand, as a *combination* of a certain type of *semantic neighborhoods* of all *concepts* used within the specified *subject domain*, as well as possibly key instances of the specified *concepts*, if there are any.

Each specific ontology of a given type is a semantic neighborhood of the corresponding (specified) subject domain. Each *ontology type* uniquely corresponds to a *subject domain*, fragments of which are specific *ontologies* of this type. Consequently, each *ontology type* has its own specialized sc-language that provides a representation of *ontologies* of this type.

ontology of the subject domain

⇒ *subdividing**:

{● *particular ontology of the subject domain*

:= [*ontology* representing the specification of the relevant subject domain in one aspect or another]

● *unified ontology of the subject domain*

:= [ontology of the *subject domain*, which is the result of combining all known *particular ontologies* of this subject domain]

}

Each *particular ontology* is a fragment of a *subject domain*, which includes all (!) particular ontologies belonging to the corresponding *ontology type*. At the same time, the specified *subject domain*, in turn, also has a corresponding *ontology*, which is no longer a meta-knowledge (like any ontology) but a meta-meta-knowledge (a specification of meta-knowledge).

particular ontology of the subject domain

⇒ *subdividing**:

{● *structural specification of the subject domain*

:= [*meta-knowledge* type describing the properties of *subject domains* corresponding to this meta-knowledge type]

:= [scheme of the subject domain]

● *set-theoretic ontology of the subject domain*

:= [sc-specification of a given subject domain within the *subject domain of sets*]

● *logical ontology of the subject domain*

:= [sc-text of the formal theory of a given subject domain]

● *terminological ontology of the subject domain*

}

structural specification of the subject domain

:= [structural ontology of the subject domain]

:= [role structure of the key elements of the subject domain]

:= [scheme of the concepts roles of the subject domain and its relation with related subject domains]

:= [scheme of the subject domain]

:= [specification of the subject domain from the point of view of graph theory and theory of *algebraic systems*]

:= [description of the internal (role) structure of the *subject domain*, as well as its external relations with other *subject domains*]

:= [description for the roles of the key elements of the subject domain (first of all, concepts), as well as the "place" of the specified subject domain in the set of similar ones]

:= [*semantic neighborhood* of the *subject domain* sign within this *subject domain* itself, which includes all *key signs* that are part of the *subject domain* (key concepts and key objects of subject domain research) with an indication of their roles (properties) within this *subject domain*, and the *semantic neighborhood* of the sign of the specified *subject domain* within the *Subject domain of subject domains*, including the relations of the specified *subject domain* with other semantically close to it *subject domains* (private and maternal, similar in one sense or another (for example, isomorphic), having the same *classes of research objects* or the same sets of *relations under study*)]

set-theoretic ontology of the subject domain

:= [*semantic neighborhood* of the specified *subject domain* within the *subject domain of sets*, describing the set-theoretic relations between *concepts* of the specified *domain*, including the relations of *relations* with their *definition areas*, and *domains*, the relations of the *parameters* used, and the classes of their *areas of definition*]

:= [ontology that describes:

□ a classification of research objects of the specified subject domain;

□ the correlation of the areas of definition and domains of the relations used with the selected classes of research objects, as well as with the selected classes of auxiliary (adjacent) objects that are not research objects in the specified subject domain;

□ a specification of the relations used and, in particular, an indication of whether all connectives of these relations are part of the specified subject domain.

]

The set-theoretic ontology of the subject domain includes:

- set-theoretic connections (including taxonomy) between all the concepts used, which are part of the specified subject domain;
- a set-theoretic specification of all *relations* that are part of the specified subject domain (orientation, arity, area of definition, domains, etc.);
- a set-theoretic specification of all parameters used in the subject domain (parameter definition areas, scales, units of measurement, reference points);
- a set-theoretic specification of all classes of structures used.

logical ontology of the subject domain

:= [formal theory of a given (specified) domain, describing various properties of concepts instances used in the specified subject domain with the help of variables, quantifiers, logical connectives, formulas]

The logical ontology of the subject domain includes:

- a formal definitions of all concepts that are definable within the specified subject domain;
- informal explanations and some formal specifications (at least examples) for all concepts that are indefinable within the specified subject domain;
- a hierarchical concepts system, in which for each concept studied in the specified subject domain, either the fact of the indefinability of this concept is indicated, or all the concepts on the basis of which the definition of this concept is given are indicated. As a result, the set of concepts under study is divided into a number of levels:
  - undefined concepts;
  - concepts of the 1st level, defined only on the basis of undefined concepts;
  - concepts of the 2nd level, defined on the basis of concepts that change the 1st level ones and below;
  - etc.
- a formal record of all axioms, i.e. propositions that do not require proof;
- a formal record of propositions whose truth requires justification (proof);
- formal texts of proving the truth of propositions, which are a specification for the sequence of steps of the corresponding reasoning (steps of logical inference, the application of various logical inference rules);
- a hierarchical system of propositions, in which for each proposition, true in relation to the specified subject domain, either the axiomaticity of this proposition is indicated, or <u>all</u> propositions are listed, on the basis of which this proposition is proved. As a result, the set of propositions that are true in relation to the specified subject domain is divided into a number of levels:
  - axioms;

  - propositions of the 1st level, proved only on the basis of axioms;
  - propositions of the 2nd level, proved on the basis of propositions that are at the 1st level and below.
- a formal record of hypothetical propositions;
- a formal description of the logical-semantic typology of propositions – propositions about existence, non-existence, uniqueness, propositions of a defining type (which can be used as definitions of the corresponding concepts);
- a formal description of various types of logical-semantic relations between propositions (for example, between an utterance and its generalization);
- a formal description of the analogy:
  - between definitions;
  - between propositions of any kind;
  - between proofs of different propositions.

terminological ontology of the subject domain
:= [ontology describing rules for constructing terms (sc-identifiers) corresponding to sc-elements belonging to the specified subject domain, as well as describing various kinds of terminological relations between the terms used, characterizing the origin of these terms]
:= [system of terms of a given subject domain]
:= [thesaurus of the relevant subject domain]
:= [dictionary of the relevant (specified) subject domain]
:= [fragment of the global *Subject domain of sc-identifiers* (external identifiers of sc-elements), providing a terminological specification of some subject domain]

Now let us take a closer look at the concept of a *unified ontology of the subject domain*.

unified ontology of the subject domain
:= [combination of all particular ontologies corresponding to the same subject domain]
⇐ *generalized combination**:
{• *structural specification of the subject domain*
• *set-theoretic ontology of the subject domain*
• *logical ontology of the subject domain*
• *terminological ontology of the subject domain*
}

subject domain and ontology
:= [integration of some *subject domain* with the corresponding *unified ontology*]
:= [subject domain & ontology]
⇐ *generalized combination**:
{• *subject domain*
• *unified ontology of the subject domain*
}
:= [sc-text that is a combination of some subject domain represented in the SC-code and the combined ontology of this subject domain, also represented in the SC-code]
:= [integration of the subject domain and all ontologies specifying this subject domain]
:= [set of various *facts* about the structure of some activity domain for some *subjects*, as well as various types of *knowledge* specifying this field of activity]
:= [facts and knowledge about a certain field of activity]
:= [sc-model of the subject domain and various onthologies specifying this subject domain (and, first of all, its key concepts) from different angles]
:= [coherent fragment of the ostis-system knowledge base from a logical and semantic point of view, focusing on a specific class of research objects and on a specific aspect of their consideration]

Subject domains and ontologies are the main *type of knowledge base sections*, having a high degree of their independence from each other and clear rules for their co-ordination, which ensures their semantic (understandable) compatibility within the entire *knowledge base*.

IX. Subject domains of the represented concepts

Each of the represented concepts corresponds to *subject domains and ontologies*, in which this concept is the maximum class of research objects:

- **Subject domain of knowledge and ostis-systems knowledge bases**

Subject domain of knowledge and ostis-systems knowledge bases
∈ *subject domain*
∋ *maximum class of research objects'*:
knowledge
∋ *class of research objects'*:
• *knowledge type*
• *relation defined on a set of knowledge*

- **Subject domain of structures**

Subject domain of structures
∈ *subject domain*
∋ *maximum class of research objects'*:
structure
∋ *class of research objects'*:
• *connected structure*

    - *disconnected structure*
    - *trivial structure*
    - *nontrivial structure*

∋ *relation under study'*:
    - *structure element'*
    - *unrepresented set'*
    - *fully represented set'*
    - *structure element that is not a set'*
    - *polymorphism**
    - *homomorphism**
    - *isomorphism**
    - *similarity of structures**

- **Subject domain of semantic neighborhoods**

Subject domain of semantic neighborhoods
∈ *subject domain*
∋ *maximum class of research objects'*:
semantic neighborhood
∋ *class of research objects'*:
    - *full semantic neighborhood*
    - *basic semantic neighborhood*
    - *specialized semantic neighborhood*
    - *terminological semantic neighborhood*
    - *explanation*
    - *note*
    - *set-theoretic semantic neighborhood*
    - *logical semantic neighborhood*

- **Subject domain of subject domains**

The *Subject domain of subject domains* includes the structural specifications of all *subject domains* that are part of the *ostis-system* knowledge base, including the *Subject domain of subject domains* itself. Thus, the *Subject domain of subject domains* is, firstly, a *reflexive set* and, secondly, a reflexive subject domain, that is, a *subject domain*, one of the research objects of which is itself.

Subject domain of subject domains
:= [Subject domain, the research objects of which are subject domains]
∈ *reflexive set*
∋ *maximum class of research objects'*:
subject domain
∋ *class of research objects'*:
    - *static subject domain*
    - *dynamic subject domain*
    - *concept*
    - *nontrivial structure*

∋ *relation under study'*:
    - *concept under study'*
    - *maximum class of research objects'*
    - *non-maximum class of research objects'*
    - *class of structures under study'*
    - *private subject domain**
    - *concept studied in the private subject domain'*

- **Subject domain of ontologies**

Subject domain of ontologies
∈ *subject domain*
∋ *maximum class of research objects'*:
ontology
∋ *class of research objects'*:
    - *structural specification of the subject domain*
    - *particular ontology of the subject domain*
    - *unified ontology of the subject domain*
    - *set-theoretic ontology of the subject domain*
    - *logical ontology of the subject domain*
    - *ontology of the subject domain*

X. Conclusion

In the article, an ontological approach to the design of knowledge bases of next-generation intelligent computer systems is considered. This approach is based on the representation of the knowledge base of intelligent computer systems based on the OSTIS Technology as a hierarchical structure of interrelated subject domains and their ontologies built on the basis of top-level ontologies.

The formal interpretation of such concepts as knowledge, structure, semantic neighborhood, subject domain, ontology has been clarified, which together made it possible to determine on their basis the ontological model of knowledge bases of next-generation intelligent computer systems.

This model can form the Kernel of the knowledge base, which will ensure the compatibility of intelligent systems due to the unified representation of knowledge.

The results obtained make it possible to increase the efficiency of the development of next-generation intelligent systems due to the component approach to the development of knowledge bases and automation tools for their development.

Acknowledgment

The author would like to thank the research group of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for its help in the work and valuable comments.

References

[1] I. Davydenko, "Ontology-based knowledge base design," *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, pp. 42–50, 2018.

[2] S. J.F, "Top-level ontological categories," *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 669–685, 1995.

[3] C. Partridge, *A survey of Top-Level Ontologies*. The Construction Innovation Hub, 2020.

[4] V. Mascardi, *A Comparison of Upper Ontologies*. DBLP, 2007.

[5] (2022, Nov) DOLCE: Descriptive Ontology for Linguistic and Cognitive Engineering. [Online]. Available: http://www.loa.istc.cnr.it/dolce/overview.html

[6] (2022, Nov) Suggested Upper Merged Ontology (SUMO). [Online]. Available: https://github.com/ontologyportal/sumo

[7] (2022, Nov) Cycorp – cycorp making solutions better. [Online]. Available: https://cyc.com/

[8] (2022, Nov) Github: Opencyc. [Online]. Available: https://github.com/asanchez75/opencyc

[9] V. Golenkov, N. Gulyakina, and D. Shunkevich, *Otkrytaja tehnologija ontologicheskogo proektirovanija, proizvodstva i jekspluatacii semanticheski sovmestimyh gibridnyh intellektual'nyh komp'juternyh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems]*. Bestprint [Bestprint], 2021.

[10] (2022, Nov) IMS.ostis Metasystem. [Online]. Available: https://ims.ostis.net

Структура баз знаний интеллектуальных компьютерных систем нового поколения: иерархическая система предметных областей и соответствующих им онтологий

Банцевич К.А.

В работе рассмотрен онтологический подход к проектированию баз знаний интеллектуальных компьютерных систем нового поколения. Данный подход основан на представлении базы знаний интеллектуальных компьютерных систем на основе Технологии OSTIS как иерархической структуры взаимосвязанных предметных областей и их онтологий, построенных на базе онтологий верхнего уровня.

Уточнена формальная трактовка таких понятий, как знание, структура, семантическая окрестность, предметная область, онтология, что в совокупности позволило определить на их основе онтологическую модель баз знаний интеллектуальных компьютерных систем нового поколения.

Данная модель может составить Ядро базы знаний, которое позволит обеспечить совместимость интеллектуальных системы за счет унифицированного представления знаний.

Полученные результаты позволяют повысить эффективность разработки интеллектуальных систем нового поколения за счет компонентного подхода к разработке баз знаний и средств автоматизации их разработки.

Received 17.11.2022

Means of formal description of syntax and denotational semantics of various languages in next-generation intelligent computer systems

Artem Goylo
Minsk State
Linguistic University
Minsk, Belarus
Email: artemgoylo@gmail.com

Sergei Nikiforov
Belarusian State University
of Informatics and Radioelectronics
Minsk, Belarus
Email: nikiforov.sergei.al@gmail.com

***Abstract*—The article is devoted to linguistic means of formal description of syntax and denotational semantics of different languages in next-generation intelligent computer systems. A formal ontology of different languages is proposed. The treatment of such notions as language, sign, information construction, sign construction, syntax, semantics, meaning, value, etc. is specified. The subject domains of syntax and denotational semantics of natural languages are formalized. As a result, an upper-level ontology is obtained, using which it becomes possible to formalize more specific subject domains of specific languages, both natural and artificial, for their further use in natural-language interfaces of next-generation intelligent computer systems.**

***Index Terms*—language, sign, information construction, natural language processing, natural language understanding, ontology, semantic network, Open Semantic Technology for Intelligent Systems (OSTIS), SC-code (Semantic Computer Code), constituency grammar**

I. Introduction

The incredible diversity of natural languages and the existence of a large number of artificial languages undermines interoperability between different people(s), different computer systems, as well as between people and computer systems.

This leads to a great number of problems connected with:

- a rapid increase in the amount of information to be processed by people for performing various kinds of tasks;
- difficulties and low efficiency of human interaction with computer systems.

These problems can be solved by designing intelligent computer systems, which are already used in automating all kinds of human activities.

Such computer systems help to solve problems that exist in the general domain of human-computer interaction, including but not limited to:

- automating the processing of such multilingual documents that cannot be processed manually in sufficient time;
- developing natural language interfaces, which are highly dependent on the quality of natural language processing;
- developing machine translation systems;

Thus, a very urgent issue is the problem of developing natural language understanding subsystems for next-generation intelligent computer systems.

In order to ensure interoperability between systems and between components of those systems, it is necessary to introduce a kind of shared foundation, upon which systems can be built. An ontology can be utilized as said foundation. Therefore, the implementation of an NLP-component for next-generation intelligent computer systems is dependent on designing a set of ontologies necessary for the component to function. The most vital ontology for natural language interfaces is the one that describes various aspects of natural languages, including their syntax and denotational semantics.

Moreover, to solve the problem of interoperability introduced at the beginning of the article, it is not enough to focus solely on natural languages. What is needed is a kind of upper ontology for languages in general and their units of meaning (which we will call information constructions). We will formalize the necessary concepts as an ontology of the subject domain of languages and information constructions.

II. State of the art

Currently, the research in Artificial Intelligence covers a wide range of problems. However, there is little compatibility in the conceptual systems of different schools, approaches, and paradigms within the domain, which results in incompatible computer systems being developed on the basis of research findings [1].

Given the complexity of modern software, it is vital to ensure that there is interoperability between different software entities, and that it is possible to re-use previous implementations in a way that they are compatible with current software.

One way to solve this problem is to create software engineering ontologies. Such ontologies should satisfy the following requirements [2]:

- formal semantics should be specified to avoid the ambiguity of definitions and the possibility of invalid interpretations, in order to ensure interoperability;

- it should be possible to apply the ontologies to a different or a more general subject domain, which would help to reduce development cost and increase the quality of the end product;
- it should be possible to perform logical inference over the ontology.

An example of an ontology in this field is COPS [3]. The aim of this ontology was to formalize general concepts in the domain of software engineering in order to simplify the development and use of software.

The issue of compatibility between research results is also of concern in linguistics – a science that has many (often incompatible) theories. Linguists utilize different annotation schemas, different approaches to structuring of corpora, and different ways of representing data in them [4], [5].

In order to solve the problem of incompatibility between annotation schemas, there have been proposed different standards for annotation formats. Among the examples of such efforts are Text Encoding Initiative – a digital data representation consortium [6] – and guidelines proposed by EAGLES (Expert Advisory Group on Language Engineering Standards), for example, – their corpus encoding recommendations [7]. However, none of the standards became widely used by linguists [8, p. 4].

Instead of providing recommendations for linguistic data annotation, a more effective way of solving the aforementioned problems has been proposed – that is, creating linguistic ontologies [9], [10]. Apart from the fact that an upper ontology for the domain of linguistics can provide a link between divergent linguistic theories, such an ontology by its nature is conceived as a formal description of concepts used in linguistics, represented in a machine-readable format, which means that it can be used in computer systems capable of understanding annotated linguistic data, performing intelligent search over language corpora, as well as, potentially, reasoning over linguistic research [4].

As a result, an ontology of the domain of linguistics has been created – The General Ontology of Linguistic Description (GOLD) [11]. This ontology describes the most basic categories and relations used in linguistics, and the ontology itself integrated into a top-level ontology called Suggested Upper Merged Ontology (SUMO) [12]. The authors of GOLD state that they created the ontology first and foremost with a view to solve the problem of interoperability of linguistic typology data so that expert systems could process scientific evidence of natural languages – that is, the ontology was not aimed at solving the problems in the domain of natural language processing *per se* [13, p. 4].

A natural language ontology, aimed at being used in natural language processing tasks is Ontologies of Linguistic Annotation (OLiA) [14]. The main idea of this ontology is to ensure compatibility between linguistic data annotations produced by computer systems while analyzing natural language texts on the one hand, and the corresponding linguistic concepts within the ontology on the other hand. As opposed to other linguistically-motivated ontology, OLiA provides not only an inventory of concepts and relations, but also specifies the way to integrate the elements of the ontology with linguistic data annotations used, for example, in corpora [14, p. 4].

When creating an ontology of natural language, it is necessary to pay attention to the status of specifications of linguistic information within such ontology. J. Bateman suggests to differentiate between three types of ontologies according to the degree with which natural language-specific information is integrated into the ontology [15]:

1) ontologies that are an abstract semantico-conceptual representation of world knowledge, which is used directly as a specification of denotational semantics for syntax and lexicon of a natural language (mixed ontologies);
2) ontologies that have a separate specification of denotational semantics of natural languages, which is used as an interface between natural language syntax and the conceptual ontology itself (interface ontologies);
3) ontologies that are an abstract specification of real world knowledge that pays no directed attention to meanings encoded in natural language (conceptual ontologies).

Especially popular within the field of natural language processing are ontologies of the second type [15, p. 8], because such ontologies (in contrast to the ontologies of the third type) make it possible to formalize more information about natural languages. For example, one of the most popular ontologies used in natural language processing, the Generalized Upper Model [16], is a second-type ontology [15]. P. Buitelaar et al. [17] stress that all formal ontologies have to be linked with linguistic information in order to solve such tasks as extracting information from natural language texts, automated population of ontologies, and natural language text generation.

Since ontological approach to NLP allows to specify the semantics of data obtained as a result of processing a text as well as to potentially raise the quality of NLP-analysis, there has been a shift towards creating ontology-driven NLP systems [18], [19]. Natural language ontologies are actively used in NL-text generation from some domain ontology [20], [21].

Ontological approach is also used in systems for making natural language queries to databases, in which a natural language query is translated into a subject domain ontology query language, with the result of the translation itself then being translated into SQL in order to facilitate user interaction with relational databases [22].

Moreover, specifying linguistic information within an ontology helps in automated ontology design based on natural language texts [23].

More specialized ontologies for specific subdomains of linguistics are being created: for example, an ontology of spacial expressions in natural languages [24], an ontology of temporal expressions [25], ontologies of individual languages [26]. When using ontologies in NLP, it is important to "link" the concepts from an ontology with the lexicon of a specific natural language. This led to the creation of extensions for popular linguistic databases, such as WordNet [27], VerbNet [28] and FrameNet [29], to use the databases together with top-level ontologies (for an example, see [30]). There is an active ongoing development of ontologies of natural language lexica, which resulted in a variety of formal descriptions of lexica being created [31], [32], [33], [34], [35]. Because popular lexical databases were not designed to serve as an ontology and do not possess the necessary degree of formalization (e.g., WordNet), ontologies that could serve as a sort of "superstructure" over such databases are being created. One such ontology that is widely used is Lemon [36].

Many of the ontologies above are designed using the Semantic Web technology [37], which possesses several disadvantages, namely:

1) there is no rigorous and at the same time simple formal basis for representing information (a kind of kernel, or a representation invariant), which would be universal in the sense that all other systems could be designed based on this kernel. The model that is de facto used in this way is RDF [38] [39], but it does not sufficiently satisfy the requirements of universality and formality.
2) no basic relations have been defined that could be reflected in the syntax of the basic language itself. All relations have to be specified explicitly.

Thus, the technology used to design such ontologies does not allow to structure information space in a formal enough way, streamline information search, ensure compatibility of descriptions provided by different authors – that is, it cannot be viewed as a universal language for representing any kinds of data in knowledge bases.

Moreover, Semantic Web is a technology that is external in relation to existing solutions for natural language processing, which is why such systems have to interact with it using APIs or standardized query languages (in particular, SPARQL) [21].

It is important to highlight the fact that, despite very active development of ontology-driven NLP systems, many popular NLP-libraries (e.g., NLTK [40] and spaCy [41]) do not support the use of ontologies, and the majority of natural language text markup tools use specific formats, which makes it necessary to use parsers and converters specific for such tools in order to utilize them in NLP-related tasks [42, p. 3].

In conclusion, the following problems in state-of-the-art approaches to natural language processing:

1) Lack of unification (standardization) in the approaches described above leads to increased costs of their integration, which significantly complicates the development of various systems on their basis [43], [1].
2) Despite the fact that ontologies can help to solve a wide range of NLP tasks, the majority of ontology driven NLP systems focus on addressing very specific problems (for example, systems can focus only on text generation, or ontology population, or natural language querying).
3) A number of specialized linguistic ontologies have been designed which only formalize a subdomain of linguistics (lexicon, in particular), which is in some sense a consequence of the problem described above. At the same time, the existing upper-level linguistic ontologies (e.g., OLiA) do not completely solve the problem of unification because they have to introduce an intermediate level of representation in order to integrate data collected by an NLP system with the corresponding ontology fragments.

The solution to these problems requires a language with enough expressive power to describe knowledge of any kind, it also requires a technology aimed directly at designing interoperable next-generation intelligent computer systems. The OSTIS technology satisfies both of the requirements. Because of this, natural language interfaces of systems designed using the OSTIS technology (*ostis-systems*) will be able to solve a wide range of NLP-related problems – be it natural language synthesis in general, dialoging using natural languages, NL-driven search, information extraction, etc. While currently such tasks are often performed by specialized tools and require additional effort to ensure potential compatibility with individual computer systems, the OSTIS Technology utilizes a single universal knowledge representation language (called *SC-code*), in which all components of the problem solver as well as the ontology of languages and ontologies of specific subject domains are implemented, which will help to solve the problem of interoperability.

Moreover, a natural language ontology designed using such a technology could be used not only in practical NLP-related applications but also to ensure interoperability of data obtained by linguistic research, which would be a valuable contribution to theoretical linguistics.

Finally, an ontology of natural language can be viewed as a subset of an ontology of languages in general (not only natural ones but also formal and artificial) – something that the aforementioned ontologies do not do. This will make it possible to conceptualize natural languages and programming languages within the same information space, and to unify the concepts used in these different domains to more efficiently solve natural language processing tasts in intelligent computer systems.

The aim of this article is to propose basic means of formal description of syntax and denotational semantics of various languages. This will be done by way of presenting a fragment of the ontology of languages and information constructions created using the OSTIS Technology, which can be used to design next-generation intelligent computer systems.

III. Ontology of languages and information constructions

As a solution to the problem of interoperability, we suggest representing knowledge about various languages (including knowledge of their syntax and semantics) in a unified way. This will help to ensure compatibility of such representations and reduce the cost of developing ontology-driven computer systems.

In our approach, we propose using *the OSTIS Technology* [43], which helps to facilitate interoperability of different problem solver models and reduce the number of modifications introduced when adding, for example, a new model of the problem solver.

Systems developed on the basis of the OSTIS Technology are called ostis-systems. The OSTIS Technology is based on a universal way of semantic representation of information in the memory of intelligent computer systems, called *SC-code*. SC-code texts are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, depending on their directivity, can be *sc-arcs* or textitsc-edges). *SC-code alphabet* consists of five main elements, on the basis of which SC-code constructions of any complexity are built, including the introduction of more specific types of sc-elements (for example, new concepts). The memory that stores SC-code constructions is called semantic memory or *sc-memory*.

Within the framework of the technology, the structure of the *knowledge base* of any ostis-system is described by a hierarchy of subject domains and their corresponding ontologies. At the same time, an ontology is treated as a specification of the corresponding subject domain.

Fragments (substructures) of the subject domains and ontologies, as well as structures related to the models of the knowledge base and problem solver, will be shown below in the form of SC-code texts (sc-texts).

In the following sections we will describe the proposed ontologies.

A. Subject domain of languages and information constructions

sign
⇒ *subdividing**:
{• *sign, which is an element of a discrete information structure*
• *sign, which is a non-atomic fragment of a discrete information structure*
}

Sign is a fragment of an information construction that conventionally represents (depicts) some describable entity, which is called the denotation of the sign.

Since all signs are discrete information constructions, the set of signs is the domain of all relations defined on the set of discrete information constructions.

the relation defined on the set of signs^
∋ *synonymy of signs**

Signs are synonymous if and only if they denote the same entity. At the same time, synonymous signs may or may not be syntactically equivalent.

sign construction
⊂ *discrete information structure*

Sign construction is a discrete information construction, which generally is a configuration of signs and special fragments of an information construction that provide structuring of the configuration of signs.

*Sign** is a binary oriented relation linking a sign construction to the set of all signs included in it.

*Semantic adjacency of sign constructions** is a binary relation linking semantically adjacent sign constructions. The sign constructions $\boldsymbol{Ti}$ and $\boldsymbol{Tj}$ are adjacent if and only if there are synonymous signs $\boldsymbol{ti}$ and $\boldsymbol{tj}$, one of which is a part of the construction $\boldsymbol{Ti}$ and the other is a part of the construction $\boldsymbol{Tj}$.

class of sign constructions^
∋ *semantically elementary sign construction*
∋ *semantically coherent sign construction*

Semantically elementary sign construction is a sign construction describing some (one) relationship between some entities.

Semantically coherent sign construction is a sign construction that can be represented as a concatenation of semantically elementary sign constructions, each of which is semantically adjacent to the preceding and subsequent semantically elementary sign construction.

parameter defined on the set of sign constructions^
∋ *semantic coherence of sign constructions^*
 ∋ *semantically coherent sign construction*
 ∋ *semantically incoherent sign construction*

Information construction is a construction (structure) containing some information about some entities. The form of representation ("image", "materialization"), the form of structuring (syntactic structure), as well as *meaning** (denotational semantics) of *information constructions* can be very different.

Discrete information construction is an *information construction* whose meaning is defined by:

- the set of elements (syntactically atomic fragments) of this information construction,
- alphabet of these elements - a family of classes of syntactically equivalent elements of the information construction,
- inclusion of each element of the information construction to the corresponding class of syntactically equivalent elements of the information construction,
- configuration of incidence relations between the elements of the information construction.

A consequence of this is that the form of representation of the elements of a discrete information construction does not need to be specified for the analysis of its meaning. It is important to ensure the following:

- the presence of a simple procedure for selecting (segmenting) elements of a discrete information construction,
- specification of a simple procedure for establishing the syntactic equivalence of different elements of a discrete information construction,
- the presence of a simple procedure for determining whether each element of a discrete information construction belongs to the corresponding class of syntactically equivalent elements (i.e. to the corresponding element of the alphabet).

Element of a discrete information construction is a syntactically atomic fragment (symbol) included in a discrete information construction. Since discrete information constructions can have common elements (atomic fragments) and even some of them can be parts of other information constructions, an element of a discrete information construction can be a part of several information constructions at once.

Next we will consider the relations defined on the set of discrete information constructions.

the relation defined on the set of discrete information constructions^

∋ *discrete information construction element**
∋ *syntactic equivalence of elements of discrete information constructions**
∋ *incidence of elements of discrete information constructions**
∋ *non-elementary fragment of a discrete information construction**
∋ *alphabet of a discrete information construction**
∋ *primary syntactic structure of a discrete information construction**
∋ *syntactic equivalence of discrete information constructions**
∋ *copy of a discrete information construction**
∋ *semantic equivalence of discrete information constructions**
∋ *semantic extension of a discrete information construction**
∋ *syntax of an information construction**
∋ *meaning**
∋ *operational semantics of an information construction**

*Element of a discrete information construction** is a binary oriented relation, each pair of which links (1) a sign of some discrete information construction and (2) a sign of one of the elements of this discrete information construction*.

*Syntactic equivalence of elements of discrete information constructions** is the relation linking syntactically equivalent elements (atomic fragments) of the same or different discrete information constructions, i.e. elements belonging to the same class of syntactically equivalent *elements of discrete information constructions**.

*Incidence of elements of discrete information constructions** for *linear information constructions* is a sequence of elements (symbols) included in these constructions. For discrete information constructions whose configuration has a non-linear nature, the incident relation of their elements can be broken down into several partial incidence relations, each of which is a subset of the combined incidence relation. For example, for two-dimensional discrete information constructions it is (1) the incidence of elements of information constructions "horizontally" and (2) the incidence of elements of information constructions "vertically".

*Elementary fragment of a discrete information construction** is a binary oriented relation linking a given discrete information construction with a discrete information construction which is a substructure for it, which includes (1) a subset of elements of the given information construction and, respectively, (2) a subset of incidence pairs of elements of the given information construction.

*Alphabet of a discrete information construction** is a binary relation linking a discrete information construction with a family of pairwise non-overlapping classes of syntactically equivalent elements of a given discrete information construction*.

*Primary syntactic structure of a discrete information construction** is a binary oriented relation linking a discrete information construction to a *graphic structure* that fully describes its configuration and which includes: (1) signs of all those classes of syntactically equivalent elements to which the elements of the described discrete information construction belong, (2) signs of all elements (atomic fragments) of the information construction described, (3) pairs describing the incidence of the elements of the information construction described, (4) pairs describing belonging of elements of the information construction described to the corresponding classes of syntactically equivalent elements of this information construction.

*Syntactic equivalence of discrete information constructions**: Discrete information constructions $\boldsymbol{Ti}$ and $\boldsymbol{Tj}$ are syntactically equivalent if and only if there exists an isomorphism between the construction $\boldsymbol{Ti}$ and the construction $\boldsymbol{Tj}$, in which each element of the construction $\boldsymbol{Ti}$ corresponds to a syntactically equivalent element of the construction $\boldsymbol{Tj}$, i.e., an element from the same class of syntactically equivalent elements of the construction $\boldsymbol{Tj}$, i.e. an element belonging to the same class of syntactically equivalent elements of discrete information constructions. And vice versa.

*Copy of a discrete information construction** is a binary oriented relation that links a discrete information construction with a discrete information construction which is not only syntactically equivalent to it, but also contains information about the form of representation of elements of this copied information construction*

copy of a discrete information structure*

⊂ *syntactic equivalence of discrete information constructions**

*Semantic equivalence of discrete information constructions**: Information construction $\boldsymbol{Ti}$ and information construction $\boldsymbol{Tj}$ are semantically equivalent if and only if each entity (including each relationship between entities) described in information construction $\boldsymbol{Ti}$ is also described in information construction $\boldsymbol{Tj}$. And vice versa.

*Semantic extension of a discrete information construction**: the information construction $\boldsymbol{Tj}$ is a semantic extension of the information construction $\boldsymbol{Ti}$ if and only if each entity described in $\boldsymbol{Ti}$ is also described in $\boldsymbol{Tj}$, but the reverse is not true.

*Syntax of an information construction** is a description of what parts a given information construction consists of and how these parts (fragments) are related to each other.

*Meaning** (*denotational semantics of an information construction**) is a binary oriented relation, each pair of which relates some information construction to its explicit (formal) representation of what entities this information construction describes and how these entities are related to each other.

*Operational semantics of an information construction** is a binary oriented relation, each pair of which connects a sign of some information construction with a set of rules of its transformation - a description of what rules can be used to perform actions on transformation (processing, transformation) of a given information construction, leaving it within the class of syntactically and semantically correct information constructions.

operational semantics of an information construction*
⇒ *second domain**:
  operational semantics of an information construction

Now let us consider mappings defined on the set of discrete information constructions.

mapping defined on the set of discrete information constructions
∋ *mapping between the syntactic structure of the information construction and the meaning of that construction**
  ⊂ *mapping**

Mapping defined on the set of discrete information constructions is the set of ordered pairs, the first component of which is an ordered pair consisting of (1) a sign of the syntactic structure of some information construction and (2) a sign of the semantic structure of that construction, and the second component of which is a set of ordered pairs linking fragments of the syntactic structure of a given information construction (which describe either the structure of fragments of a given construction or links between fragments of this construction) with those fragments of the semantic structure of a given information construction that are semantically equivalent to either syntactically represented fragments of a given information construction or syntactically represented links between such fragments.

Discrete information constructions have the following parameters.

parameter defined on the set of discrete information constructions^
∋ *dimensionality of discrete information constructions^*
  := [typology of discrete information constructions based on their dimensionality]
  ∋ *linear information structure*
  ∋ *two-dimensional information structure*
  ∋ *three-dimensional information design*
  ∋ *four-dimensional information structure*
  ∋ *graph information structure*

Linear information construction is a discrete information construction, each element of which can have no more than two incident elements (one on the left and one on the right).

Two-dimensional information construction is a discrete information construction, each element of which can have no more than four incident elements (left-right, top-bottom).

Three-dimensional information construction is a discrete information construction, each element of which can have no more than six incident elements (left-right, top-bottom, back-to-front).

Four-dimensional information construction is a discrete information construction, each element of which can have no more than eight incident elements (for example, left-right, top-bottom, front-back, earlier-later).

Graph information construction is a discrete information construction whose set of elements is divided into two subsets - sheaves and nodes. In this case nodes can have an unlimited number of incident sheaves. In some graph information constructions sheaves can have an unlimited number of other sheaves incident to them.

parameter defined on the set of discrete information constructions^
∋ *typology of discrete information constructions, defined by their carrier^*
  ∋ *non-computer form of representation of discrete information constructions*
    ⊃ *audio message*
    ⊃ *an information construction presented in sign language*
    ⊃ *information construction presented in written form*
  ∋ *file*

Representation of information constructions in the form of *files* is directed at representation of discrete (!) information constructions. Therefore "file" representation of non-discrete information constructions (for example, various kinds of signals) implies "discretization" of such constructions, i.e. their transformation into discrete ones. This is how audio signals (in particular, speech messages), images, video signals, etc. are converted.

parameter defined on the set of discrete information constructions^
∋ *level of unification of representation of syntactically equivalent elements of discrete information constructions^*
  ∋ *discrete information construction with a low level of unification of the representation of elements*
    ⊃ *audio message*
    ⊃ *an information construction presented in sign language*
    ⊃ *manuscript or a copy thereof*

∋ *discrete information construction with a high level of unification of element representation*
⊃ *printed text*
⊃ *file*

*Level of unification of representation of syntactically equivalent elements of discrete information constructions^* is the level of "articulateness" of discrete information constructions.

The higher the level of unification of the representation of the elements of discrete information constructions, the easier it is to implement:

- the procedure for segmenting the elements of a discrete information construction,
- the procedure for establishing syntactic equivalence of these elements,
- the procedure for their recognition, i.e. the procedure of establishing their belonging to the corresponding classes of syntactically equivalent elements.

Having clarified the notions of sign, sign construction, information construction, discrete information construction and having considered the corresponding relations, we can proceed to the formalization of the concept of "language".

Language is a class of sign constructions, for which there are (1) general rules of their construction and (2) general rules of their correlation with those entities and configurations of entities, which are described (reflected) by the specified sign constructions.

*A relation defined on a set of languages^* is a relation whose domain includes a set of all possible languages.

*The text of a given language** is a binary relation linking the language and a syntactically correct (well-formed) sign construction of that language.

*Syntactically correct sign construction for a language** is a binary relation linking the language and a sign construction that does not contain syntactic errors for that language.

a relation defined on the set of languages^
:= [relation, the domain of which includes the set of all possible languages]
∋ *text of a language**
= (*syntactically correct sign construction for a given language** ∩ *syntactically complete sign construction for a given language*)
∋ *syntactically correct sign construction for a language**
∋ *syntactically complete sign construction for a language**
∋ *syntactically incorrect sign construction for a language**
= (*syntactically incorrect sign construction for a language** ∪ *syntactically incoherent sign construction for a language**)
⊃ *syntactically incorrect sign construction for a language**
⊃ *a syntactically incoherent sign construction for a language**
∋ *knowledge presented in a language**
:= [semantically correct text of a language*]
= *(semantically correct text of a language** ∩ *semantically complete text of a language*)*
∋ *semantically correct text of a language**
:= [text of a given language that does not contain semantic errors that contradict recognized patterns and facts*]
∋ *semantically coherent text of a language**
:= [the text of a given language containing sufficient information to establish its truth*]
∋ *semantically incorrect text for a language**
= *(semantically incoherent text for a language** ∪ *semantically incoherent text for a language*)*
⊃ *semantically incorrect text for a language**
⊃ *semantically incoherent text for a language**
∋ *alphabet**
:= [the alphabet of a given information construction or a given language*]
:= [family of classes of syntactically equivalent elements (elementary fragments) of a given information construction or information constructions of a given language*]
∋ *language**
:= [is a theory of well-formed information constructions belonging to a given language*]
:= [the syntactic rules of a given language*]
:= [Binary oriented relation, each pair of which connects a sign of some language with the description of syntactically distinguished classes of fragments of constructions of the given language, with the description of relations defined on these classes and with the conjunction of quantized statements, which are syntactic rules of the given language, i.e. rules, which all syntactically correct (well-formed) constructions (texts) of the given language should satisfy*.]
⇒ *second domain**:
language syntax
∋ *a description of the syntactic concepts of the language**
:= [description of syntactically distinguishable classes of fragments of constructions of a given language*]
⇒ *second domain**:
a description of the syntactic concepts of the language
⇐ *generalized inclusion**:
language syntax
∋ *syntactic rules of the language**
:= [the syntactic rules of a given language*]
⇒ *second domain**:
syntactic rules of the language
∋ *denotational semantics of a language**
:= [is a theory of morphisms, linking well-formed information constructions of a given language with described configurations of described

entities*]

∋ *denotational semantics of language**

:= [semantic rules of a given language*]

:= [be the semantic rules of a given language*]

:= [A binary oriented relation, each pair of which connects a sign of some language with a description of basic semantic notions of a given language and a conjunction of quantifier statements, which are semantic rules of a given language, i.e. the rules to which semantically correct meaningful information constructions corresponding (semantically equivalent) to syntactically correct constructions (texts) of a given language should satisfy*]

⇒ *note**:
[When formulating the semantic rules of a given language, concepts introduced in basic ontologies (top-level ontologies) are used.]

⇒ *second domain**:
denotational semantics of language

∋ *description of the semantic concepts of a language**

⇒ *second domain**:
description of the semantic concepts of a language

∋ *semantic rules of a language**

⇒ *second domain**:
semantic rules of a language

∋ *semantic equivalence of languages**

:= [be semantically equivalent languages*]

⇒ *определение**:
[Language ***Li*** and language ***Lj*** will be considered *semantically equivalent languages** if and only if for every text belonging to language ***Li***, there is a *semantically equivalent text** belonging to language ***Lj*** and vice versa.]

∋ *semantic extension of a language**

⇔ *inverse relation**:
*semantic reduction of a language**

⇒ *definition**:
[Language ***Lj*** will be considered a *semantic extension** of language ***Li*** if and only if for every text belonging to language ***Li*** there is a *semantically equivalent text** belonging to language ***Lj***, but the reverse is not true.]

∋ *syntactic extension of a language**

:= [be a semantically equivalent superset of a given language*]

⇒ *definition**:
[The language ***Lj*** will be considered a *syntactic extension** of ***Li*** if and only if

□ $Lj \supset Li$ (that is, all texts of ***Li*** are also texts of ***Lj***, but the reverse is not true);

□ Language ***Lj*** and language ***Li*** are *semantically equivalent languages**.

]

∋ *syntactic core of a language**

:= [be the syntactic core of a given language*]

:= [be a semantically equivalent subset of a given language with minimal syntactic complexity*]

∋ *the direction of the syntactic extension of the kernel of a given language**

:= [be the rule of transformation of information constructions belonging to a given language, which describes one of the directions of transition from the set of constructions of the core of this language to the set of all information constructions belonging to it*.]

∋ *operational semantics of a language**

:= [A binary oriented relation, each pair of which associates a sign of some language with a set of rules for transforming texts of that language*.]

⇒ *second domain**:
operational semantics of the language

∋ *internal language**

:= [be an internal language for a given information processing-based system or a given set of such systems*.]

:= [be the language of the internal representation of information in the memory of a given information processing-based system or a given class of such systems*.]

∋ *external language**

:= [be an external language for a given information processing-based system or a given set of such systems*.]

:= [be a language used to exchange information of a given information processing-based system, or a given set of such systems, with other information processing-based systems (including those of their own kind)*]

∋ *language used**

= (*internal language** ∪ *external language**)

:= [the language used by a given system based on information processing or a given set of such systems*.]

:= [the language spoken by the system in question, which is based on information processing]

∋ *languages used**

*Parameter defined on a set of languages^* is a family of language equivalence classes, interpreted in the context of some property (characteristic) inherent to the languages.

parameter defined in the set of languages^

∋ *semantic power of language^*

:= [a class of languages that are semantically equivalent to each other]

∋ *universal language*

:= [a class of all possible universal languages]

⇒ *note**:
[Obviously, all universal languages (if they really are, and not just claim to be) are semantically equivalent to each other, i.e. have the same semantic power.]

∋ *the level of syntactic complexity of the representation of the signs in the texts of the language^*
  ∋ *a language in whose texts all signs are represented syntactically by elementary fragments*
  ∋ *a language in whose texts signs are generally represented by syntactically non-elementary fragments*

∋ *the use of delimiters and terminators in language texts^*
  ∋ *language that does not use delimiters and terminators in its texts*
  ∋ *language that uses delimiters and terminators in its texts*

∋ *the level of complexity of the procedure for establishing synonymy of signs in language texts^*
  ∋ *language, within each text of which there are no synonymous signs*
    ⇒ *explanation**:
    [In texts of such language, the sign of each entity being described is present only once.]
  ∋ *язык, в рамках которого синонимичные знаки представлены синтаксически эквивалентными фрагментами текстов*
  ∋ *inflected language*
    := [language, within which synonymous signs can be represented by syntactically non-equivalent fragments, but by fragments which are modifications of some "kernel" of these fragments (in the declension and conjugation of these signs).]
  ∋ *a language in which synonymous signs can generally be represented by syntactically non-equivalent text fragments whose structure is unpredictable*

∋ *presence of homonymy in the texts of a language^*
  ∋ *a language whose texts contain homonymic signs*
    := [language whose texts contain syntactically equivalent, non-synonymous signs]
  ∋ *language, in the texts of which there is no homonymy of signs*

semantically distinguishable class of discrete information constructions
∋ *syntactic structure of the information construction*
  ⇐ *second domain**:
  *syntax of the information construction**
  ⊃ *primary syntactic structure of the information structure*
  ⊃ *secondary syntactic structure of the information structure*
∋ *language syntax*
∋ *a description of the syntactic concepts of a language*
∋ *syntactic rules of a language*
∋ *denotational semantics of a language*
∋ *description of the semantic concepts of a language*
∋ *semantic rules of a language*
∋ *the operational semantics of the language*
∋ *meaning*
  ⇐ *second domain**:
  *meaning**

Meaning - an explicit (formal) representation of the described entities and the relationships between them. The explicit representation of the described entities and the relationships between them requires a significant simplification of the syntactic structure of information constructions.

Ostis-system language is the language for representing information constructions in ostis-systems.

ostis-system language
⊂ *formal language*
⊂ *universal language*
⇐ *languages used**:
ostis-system
∋ *SC-code*
  := [Semantic Computer Code]
  ⇐ *internal language**:
  ostis-system
  ∈ *universal language*

To formally describe various kinds of languages, including the languages we are considering (SCg-code, SCs-code, SCn-code) a number of metalanguage notions are used.

Here are some of them: *identifier*, *class of syntactically equivalent identifiers*, *name*, *simple name*, *expression*, *external identifier**, *alphabet**, *delimiters**, *terminators**, *sentences**

The syntax of *knowledge representation languages in ostis-systems* can be formally described in various ways. For example, it is possible to use Bacus-Naurus meta-language to describe the syntax of SCs-code or its extension to describe the syntax of SCn-code. However, it is much more logical and expedient to describe the syntax of all forms of external sc-text mapping using SC-code itself. This approach will allow ostis-systems to independently understand, analyze and generate texts of the specified languages on the basis of principles common to any form of external representation of information, including non-linear ones.

*Alphabet** is a binary relation linking a set of texts to a family of maximal sets of syntactically identical elementary (atomic) text fragments belonging to a given set of texts.

Identifier is a structured sign of the corresponding (denoted) entity, which is most often a string – the name of the corresponding entity. In formal texts (including texts of SC-code, SCg-code, SCs-code, SCn-code) the main identifiers used should not be homonymic, i.e. they should unambiguously correspond to the entities being identified. Consequently, each pair of identifiers having the same structure must denote the same entity.

identifier
⊃ *sc.s-identifier*

name
⊂ *identifier*
:= [string identifier]
:= [identifier, which is a string (chain) of characters]
⇒ *action decomposition**:
{• *simple name*
:= [atomic name]
:= [a name that does not include other names]
• *expression*
:= [non-atomic name]
}
⊃ *sc.s-identifier*

*An external identifier** is a binary oriented relation, each sheaf (sc-arc) of which links some element to a file, the content of which is an external identifier (most often, a name) corresponding to the specified element. The notion of external identifier is a relative notion and important for ostis-systems because internal representation of information (in the form of SC-code texts) operates not with identifiers of described entities, but with signs whose structure does not matter.

B. Subject domain of files, external information constructions and external languages of ostis-systems

There are several languages for the external representation of sc-texts [43]:

- SCn-code;
- SCs-code;
- SCg-code.

SCg-code is an *external language** of ostis-systems, the texts of which are graph structures of a general form with precisely defined *denotational semantics**.

SCg-code
∈ *ostis-system language*
:= [Semantic Code graphical]
⇐ *external language**:
ostis-system
∈ *universal language*

SCs-code is an *external language** of ostis-systems, the texts of which are strings (chains) of characters.

SCs-code
∈ *ostis-system language*
:= [Semantic Code string]
⇐ *external language**:
ostis-system
∈ *universal language*

SCn-code is an *external language** of ostis-systems whose texts are two-dimensional matrices of symbols, which are the result of formatting of two-dimensional structuring of SCs-code texts.

SCn-code
∈ *ostis-system language*
:= [Semantic Code natural]
⇐ *external language**:
ostis-system
∈ *universal language*

To implement *ostis-systems knowledge bases*, all kinds of languages are used: *universal languages* as well as *specialized languages*, both *formal languages* and *natural languages*, both *internal languages* that provide the representation of information in the *ostis-systems* memory, as well as *external languages* providing the representation of input or output information. *Natural languages* are used exclusively to represent *files* stored in the *ostis-system* memory and formally specified within the *knowledge base* of that *ostis-system*.

In order to operate *intelligent computer systems* built on the basis of *SC-code*, besides the method of abstract internal representation of knowledge bases (*SC-code*), several methods of external representation of abstract *sc-texts* will be required, which are user-friendly and used in the design of source texts of *knowledge bases* of said intelligent computer systems and source texts of fragments of those *knowledge bases*, as well as used to display various fragments of *knowledge bases* to users according to user requests. The aforementioned external ostis-system languages (*SCg-code*, *SCs-code* and *SCn-code*) are proposed as such methods of external display of *sc-texts*.

All the main external formal languages used by ostis-systems (*SCg-code*, *SCs-code*, *SCn-code*) are different variants of the external representation of texts written in the internal language of ostis-systems - SC-code. These languages are universal and, therefore, *semantically equivalent languages**.

Moreover, each ostis-system can acquire the ability to use any external language (both universal and specialized, both natural and artificial) if the syntax and denotational semantics of that language are described in the memory of the ostis-system in its internal language (SC-code).

C. Formalization of natural languages

As explained above, in order to utilize insights from linguistics in the design of intelligent computer systems it is necessary to represent linguistic knowledge in a formal way. In this section, we propose a formalization of the basic concepts in linguistics made in a formal knowledge representation language – SC-code.

```
language
⇒   subdividing*:
    {•  natural language
        ⇒   explanation*:
            [A natural language is a language that
             was not created purposefully.]
     •  artificial language
        ⇒   explanation*:
            [An artificial language is a language
             specially designed to achieve certain
             goals.]
        ∋   Esperanto
        ∋   Python
        ⊃   constructed language
            ⇒   explanation*:
                [A constructed language is an
                 artificial language designed for
                 human communication.]
            ∋   Esperanto
    }
⊃   international language
    ⇒   explanation*:
        [An international language is a natural or arti-
         ficial language used by people from different
         countries to communicate.]
    ∋   English
    ∋   Russian
```

```
planned language
⇐   intersection*:
    {•  constructed language
     •  international language
    }
```

```
language of communication
⇐   union*:
    {•  natural language
     •  constructed language
    }
∋   English
∋   Russian
∋   Esperanto
⇐   union*:
    {•  isolating language
        ⇒   explanation*:
            [An isolating language is a language
             that is characterized by the complete
             absence of inflection and the presence
             of grammatical significance of the order
             of words consisting only of the root.]
        ∋   Chinese
     •  agglutinative language
        ⇒   explanation*:
            [An agglutinative language is character-
             ized by a developed system of affixes
             added to the invariable stem of the word,
             which are used to express the categories
             of number, case, gender, etc.]
        ∋   Japanese
     •  inflected language
        ⇒   explanation*:
            [An inflected language is characterized
             by a developed use of endings to express
             the categories of gender, number, case,
             a complex system of verb declension,
             alternation of vowels in the root, as well
             as a strict distinction between parts of
             speech.]
        ∋   Russian
     •  polysynthetic languages
        ⇒   explanation*:
            [A polysynthetic language is a language
             that relies on affixes to encode all (or
             almost all) grammatical meanings.]
    }
```

1) Formalization of natural language lexicon: A *lexeme* is a minimal unit of a language that has a semantic interpretation and denotes a concept that reflects the view of the world of a certain linguistic community [44].

A *grammatical category* is a system of oppositions between grammatical forms with homogeneous meanings. As part of our formalization, it is proposed to represent grammatical categories as classes of role relations, each of which corresponds to a certain grammatical meaning. We will list here only a fragment of the ontology that describes the most basic grammatical categories and relations.

```
grammatical category
∋   person
    ⇐   set of subsets*:
        role relation
    ∋   first person'
    ∋   second person'
    ∋   third person'
∋   number
    ⇐   set of subsets*:
        role relation
    ∋   singular number'
    ∋   plural number'
    ∋   dual number'
    ∋   trial number'
    ∋   paucal number'
∋   gender
    ⇐   set of subsets*:
        role relation
    ∋   masculine gender'
    ∋   neuter gender'
    ∋   feminine gender'
∋   case
    ⇐   set of subsets*:
        role relation
```

∋ *nominative case'*
∋ *genitive case'*
∋ *dative case'*
∋ *accusative case'*
∋ *instrumental case'*
∋ *prepositional case'*
∋ *vocative case'*
∋ *absolutive case'*
∋ *ergative case'*

∋ *tense*
⇐ *set of subsets**:
role relation
∋ *present tense'*
∋ *past tense'*
∋ *future tense'*

∋ *mood*
⇐ *set of subsets**:
role relation
∋ *indicative mood'*
∋ *imperative mood'*
∋ *irrealis mood'*

∋ *voice*
⇐ *set of subsets**:
role relation
∋ *active voice'*
∋ *passive voice'*
∋ *middle voice'*
∋ *reflexive voice'*
∋ *reciprocal voice'*

∋ *aspect*
⇐ *set of subsets**:
role relation
∋ *perfective aspect'*
∋ *imperfective aspect'*
∋ *common aspect'*
∋ *progressive aspect'*
∋ *perfect aspect'*

∋ *degree of comparison*
⇐ *set of subsets**:
role relation
∋ *positive degree of comparison'*
∋ *comparative degree of comparison'*
∋ *superlative degree of comparison'*

An example of the the way some of the above relations are formalized in the sc.g-language is shown in Figure 1.

Part of speech is a grammatical category that represents a class of syntactically equivalent natural language signs.

part of speech
⇐ *set of subsets**:
lexeme
∋ *noun*
∋ *adjective*
∋ *verb*
∋ *adverb*
∋ *preposition*
∋ *complementizer*
∋ *auxiliary*
∋ *determiner*

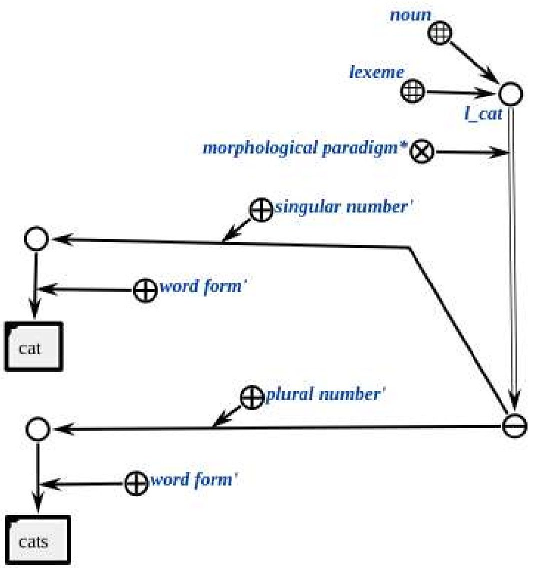


Figure 1. An example of a lexeme specification in the knowledge base

*Morphological paradigm** is a binary oriented relation connecting a lexeme and a set of its word forms.

A *word form* – a subset of a lexeme that contains all tokens of a lexeme that share certain grammatical meanings. Within our ontology, a word form is understood somewhat differently than is customary in linguistics, since all tokens of a lexeme in the OSTIS technology are considered to be instances of files.

2) Formalization of natural language syntax: When formalizing the core of natural language syntax, we drew from the framework of generative linguistics [45], [46], [47], [48].

The *distribution of a sign* is a subset of the syntactic environments that the sign appears in.

A *constituent* is an element of the set C of subsets of the tuple of tokens S, which contains as elements both S and all tokens in S in such a way that any two subsets included in C either do not intersect, or one of them is included into another.

An *immediate constituent*: let S be a set of constituents such that it includes constituents A and B. B is an immediate constituent of A if and only if B is a constituent of A and there is no constituent C such that C is a constituent of A and B is a constituent of C.

An *ultimate constituent* is an element U of a tuple of tokens T such that U is an immediate constituent in a set of constituents C and U itself has no immediate constituents.

A *phrase* is a class of constituents, which includes constituents with heads of the same part of speech. Phrases are either singletons (minimally including a single head) or an ordered pair consisting of a head and another phrase.

A *head* is a constituent whose distribution is equivalent to the distribution of the entire phrase.

constituent

⇒ *subdividing**:

{• *phrase*

• *head*

}

phrase

⇒ *subdividing**:

{• *noun phrase*

⇒ *explanation**:

[A *noun phrase* is a phrase headed by a noun.]

• *verb phrase*

⇒ *explanation**:

[A *verb phrase* is a phrase headed by a verb.]

• *adjective phrase*

⇒ *explanation**:

[An *adjective phrase* is a phrase headed by an adjective.]

• *adverb phrase*

⇒ *explanation**:

[An *adverb phrase* is a phrase headed by an adverb.]

• *prepositional phrase*

⇒ *explanation**:

[A *prepositional phrase* is a phrase headed by a preposition.]

• *complementizer phrase*

⇒ *explanation**:

[A *complementizer phrase* is a phrase headed by a complementizer.]

• *tense phrase*

⇒ *explanation**:

[A *tense phrase* is a phrase headed by an auxiliary or a modal verb.]

• *determiner phrase*

⇒ *explanation**:

[A *determiner phrase* is a phrase headed by a determiner.]

}

⇒ *subdividing**:

{• *maximal projection of a head*

• *intermediate projection of a head*

}

More specific classes can be inferred as a result of intersection between the sets of classes listed above, e.g. *maximal projection of a determiner phrase head*

maximal projection of a determiner phrase head

⇐ *intersection**:

{• *determiner phrase*

• *maximal projection of a head*

}

An example of syntactic structure of a sentence, parsed using the concepts above is shown in Figure 2.

Phrase structure is not arbitrary – elements within a phrase can only border certain sets of elements. The possible structures of phrases are given below. The sign "->" should be read as "consists of". Optional elements are shown in parentheses.

Determiner phrase:

- Maximal projection of a determiner phrase head -> (Maximal projection of a determiner phrase head) Intermediate projection of a determiner phrase head
- Intermediate projection of a determiner phrase head -> Determiner phrase head (Maximal projection of a noun phrase head)

Noun phrase:

- Maximal projection of a noun phrase head -> (Maximal projection of a determiner phrase head) Intermediate projection of a noun phrase head
- Intermediate projection of a noun phrase head -> (Maximal projection of an adjective phrase head) Intermediate projection of a noun phrase head OR Intermediate projection of a noun phrase head (Maximal projection of a prepositional phrase head)
- Intermediate projection of a noun phrase head -> Noun phrase head (Maximal projection of a prepositional phrase head)

Verb phrase:

- Maximal projection of a verb phrase head -> Intermediate projection of a verb phrase head
- Intermediate projection of a verb phrase head -> Intermediate projection of a verb phrase head (Maximal projection of a prepositional phrase head) OR Intermediate projection of a verb phrase head (Maximal projection of an adverb phrase head)
- Intermediate projection of a verb phrase head -> Verb phrase head (Maximal projection of a noun phrase head)

Adverb phrase:

- Maximal projection of an adverb phrase head -> Intermediate projection of an adverb phrase head
- Intermediate projection of an adverb phrase head -> (Maximal projection of an adverb phrase head) Intermediate projection of an adverb phrase head
- Intermediate projection of an adverb phrase head -> Adverb phrase head (Maximal projection of a prepositional phrase head)

Adjective phrase:

- Maximal projection of an adjective phrase head -> Intermediate projection of an adjective phrase head
- Intermediate projection of an adjective phrase head -> (Maximal projection of an adverb phrase head) Intermediate projection of an adjective phrase head
- Intermediate projection of an adjective phrase head -> Adjective phrase head (Maximal projection of a prepositional phrase head)

Prepositional phrase:

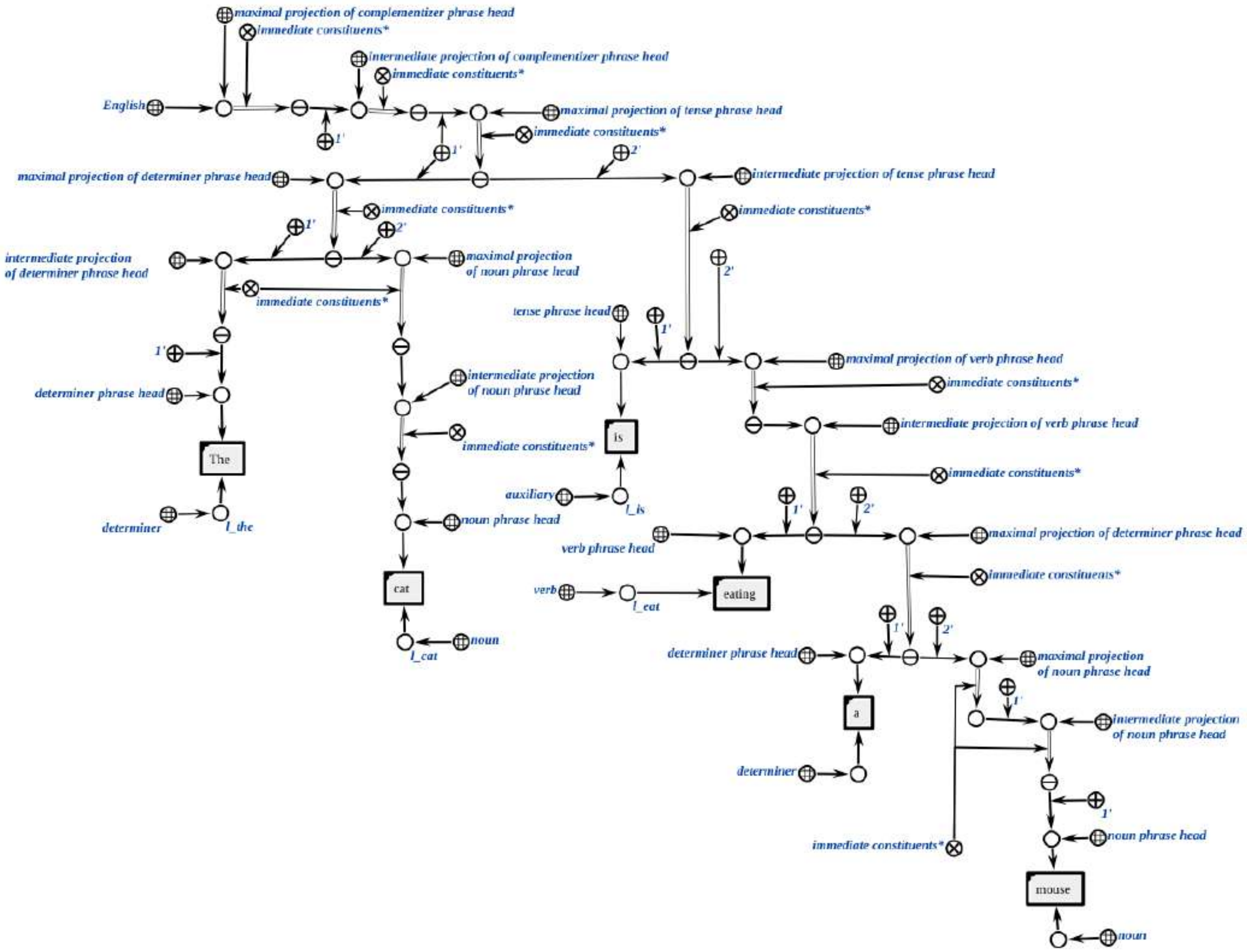

maximal projection of complementizer phrase head
immediate constituents*
intermediate projection of complementizer phrase head
immediate constituents*
English
1'
maximal projection of tense phrase head
immediate constituents*
1'
2'
maximal projection of determiner phrase head
intermediate projection of tense phrase head
immediate constituents*
immediate constituents*
1'
2'
intermediate projection of determiner phrase head
maximal projection of noun phrase head
immediate constituents*
2'
tense phrase head
1'
1'
maximal projection of verb phrase head
immediate constituents*
determiner phrase head
intermediate projection of noun phrase head
intermediate projection of verb phrase head
The
immediate constituents*
is
immediate constituents*
auxiliary
I_is
noun phrase head
determiner
I_the
1'
2'
verb phrase head
maximal projection of determiner phrase head
immediate constituents*
verb
I_eat
eating
cat
noun
I_cat
1'
2'
determiner phrase head
maximal projection of noun phrase head
a
1'
intermediate projection of noun phrase head
determiner
1'
immediate constituents*
noun phrase head
mouse
noun

Figure 2. Syntactic structure example.

- Maximal projection of a prepositional phrase head -> Intermediate projection of a prepositional phrase head
- Intermediate projection of a prepositional phrase head -> Intermediate projection of a prepositional phrase head (Maximal projection of a prepositional phrase head) OR (Maximal projection of an adverb phrase head) Intermediate projection of a prepositional phrase head
- Intermediate projection of a prepositional phrase head -> Prepositional phrase head (Maximal projection of a noun phrase head)

Tense phrase:

- Maximal projection of a tense phrase head -> (Maximal projection of a determiner phrase head) Intermediate projection of a tense phrase head
- Intermediate projection of a tense phrase head -> Tense phrase head (Maximal projection of a verb phrase head)

Complementizer phrase:

- Maximal projection of a complementizer phrase head -> (Maximal projection of some phrase head) Intermediate projection of a complementizer phrase head
- Intermediate projection of a complementizer phrase head -> Complementizer phase head Maximal projection of a tense phrase head

These rules can be formalized as follows (Figure 3).

A *complement* is a phrase that is a sister of a head. Sisters are constituents that are in an immediate constituent relation with the same constituent.

An *adjunct* is a phrase that is a daughter (immediate constituent) of an intermediate projection and a sister of the intermediate projection of the head of the same phrase.

A *specifier* is a phrase that is a daughter of a maximal projection and a sister of an intermediate projection.

The phrase structure rules can be generalized and reduced to three more abstract ones.

Specifier rule: XP -> (YP) X'

Adjunct rule: X' -> X' (ZP) | X' -> (ZP) X'

Complement rule: X' -> X (WP)

These rules are formalized in the same manner as in Figure 3.

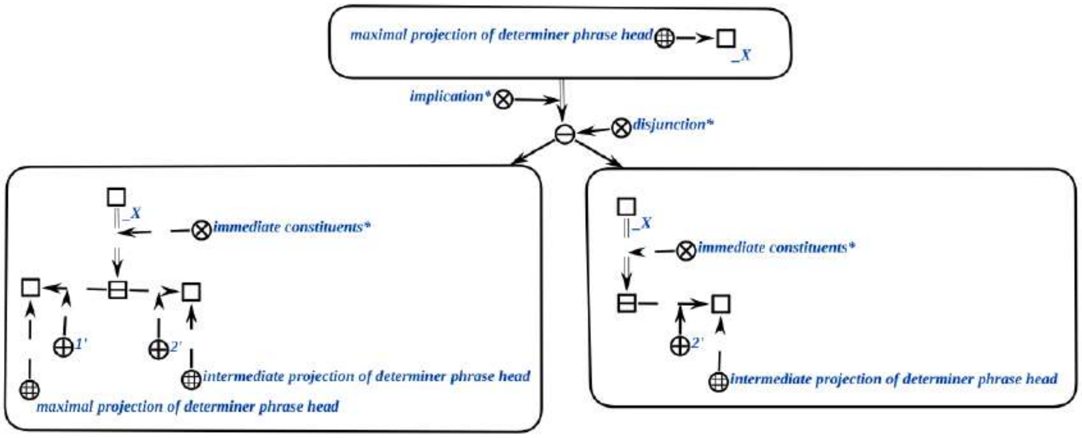

maximal projection of determiner phrase head
_X
implication*
disjunction*
_X
immediate constituents*
1'
2'
intermediate projection of determiner phrase head
maximal projection of determiner phrase head
_X
immediate constituents*
2'
intermediate projection of determiner phrase head

Figure 3. Phrase structure rule example.

3) Formalization of natural language denotational semantics: The denotational semantics of a language specifies the interpretation of the syntactic elements of that language and is a set of formulas that describe how the sign constructions of the language are associated with the entities they denote and the configurations of relations between these entities.

The denotational semantics of natural languages must be compositional, i.e. the interpretation of an entire construction must be derived from the interpretation of its individual parts. Thus, it is necessary to provide a formal description of the interpretation of NL syntax elements presented in the previous section, as well as a description of the rules for combining the interpretation of individual elements to obtain the meaning of the entire construction.

In this article, we propose a variant of the formalization of the denotational semantics of natural languages within the framework of the OSTIS technology, for which the ideas of model-theoretic formal semantics [49], [50], [51] were used.

Below are examples of rules that implement the denotational semantics of English. The rules must be applied one after the other and allow us to obtain the meaning of a natural language sentence from its syntactic structure, "climbing" the syntactic tree from heads to maximal projections.

Figure 4 shows the rule that is used to interpret heads of noun phrases and adjective phrases. We assume that the meaning of such heads is a class of elements (written here in the sc.g language), e.g. the adjective "black" is associated with a corresponding set of black objects, while the noun "cat" is associated with a set of cats. The construction to the right specifies that an entity X (represented by a variable sc.g-node) is linked via a *meaning** relation to some struct that includes a class, i.e. this entity X is a class.

Figure 5 shows the rule of interpreting a noun phrase, the maximum projection of which also includes an adjective phrase. As mentioned above, to apply this rule, you must first apply the rule shown in figure 4. The meaning of such constructions is a class that is an intersection of classes obtained as a result of interpreting the heads of the adjective and noun phrases separately. For example: the meaning of "black cat" is a set of black cats, i.e. the intersection of a set of cats and a set of black objects.

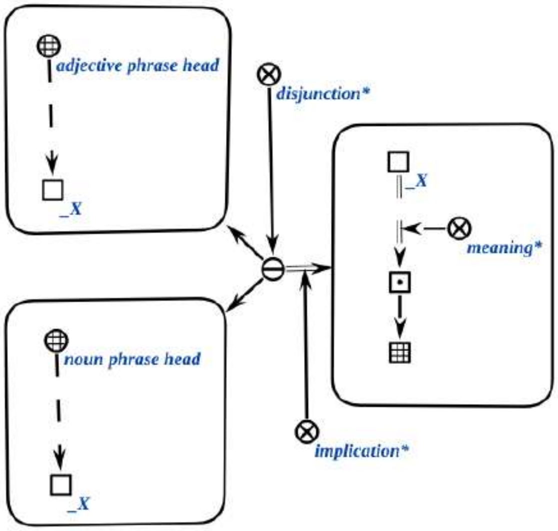


Figure 4. Noun phrase head and adjective phrase head semantic interpretation rule.

Figure 6 shows the rule according of interpreting a verb phrase. We need to include the entire branch of the verb phrase in the premise of the rule because this is how we can determine the type of the verb – this rule is intended for the interpretation of intransitive verbs. The meaning of this construction is a class of actions.

Figure 7 shows the rule of interpreting a determiner phrase headed by an indefinite article. The meaning of such a construction is the existence of an element of the class that corresponds to the meaning of the noun phrase included in this determiner phrase.

Figure 8 shows the rule of interpreting an intermediate projection of a tense phrase head, consisting of an auxiliary verb and a full verb. The auxiliary verb in this case specifies

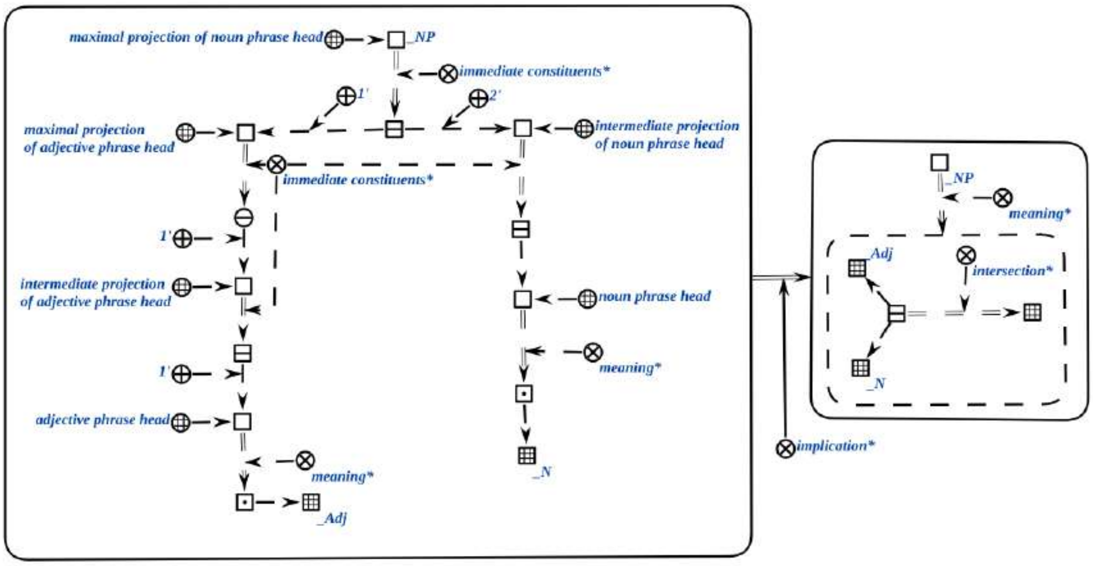


Figure 5. Semantic interpretation rule for maximal projections of noun phrase heads.

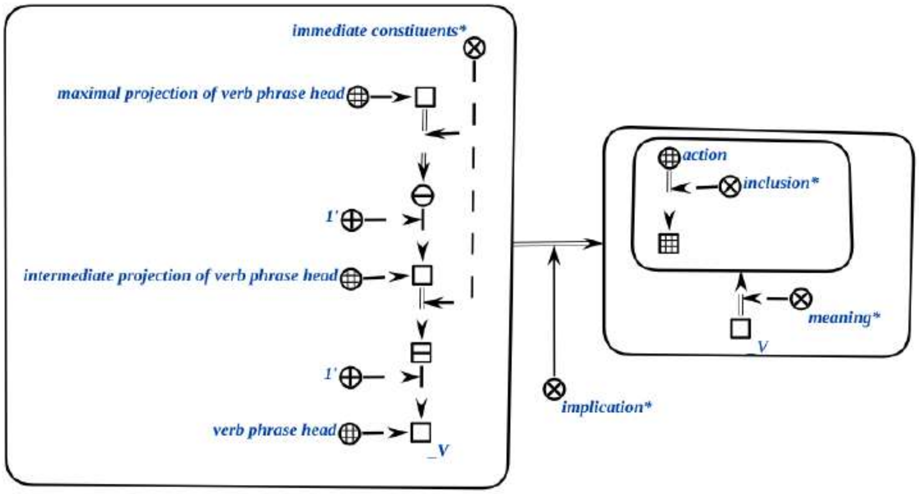


Figure 6. Semantic interpretation rule for maximal projections of intransitive verbs.

the class of actions with respect to time (whether it is planned, in progress, already completed, etc.).

Figure 9 shows the rule of interpreting a maximal projection of a tense phrase head. In this case, the maximal projection includes the subject, represented by the determiner phrase in the specifier position. The resulting meaning is a combination of the meaning of the determiner phrase obtained at an earlier stage of analysis and the meaning of a tense phrase. The determiner phrase is interpreted as the subject of the action denoted by the verb.

Figure 10 shows the rule of interpreting a maximal projection of a complementizer phrase head. This rule specifies the interpretation of a sentence with a transitive verb and is obtained as a result of applying all previously listed rules.

IV. Conclusion

In this article the linguistic means of formal description of syntax and denotational semantics of different languages in intelligent computer systems of the new generation have been described. A formal ontology of various languages has been proposed. The treatment of such notions as language, sign, information construction, sign construction, syntax, semantics, meaning, value, etc. has been clarified. A formalization of the subject domains of syntax and denotational semantics of natural languages has been proposed.

All developed means of describing the syntax and denotational semantics of languages are presented in a unified form, which allows to ensure their compatibility and significantly

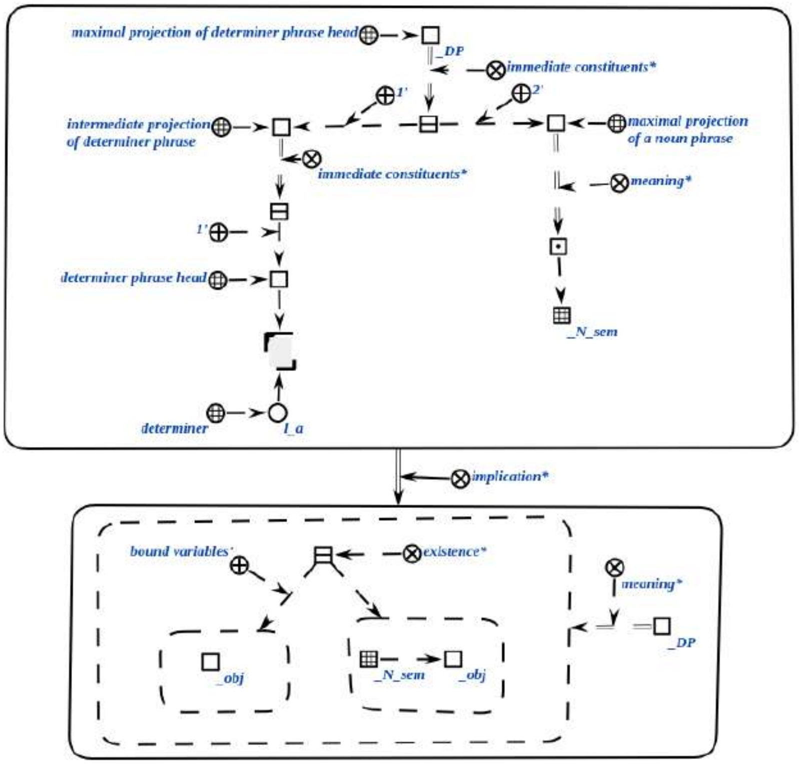


Figure 7. Semantic interpretation rule for maximal projections of determiner phrase heads.

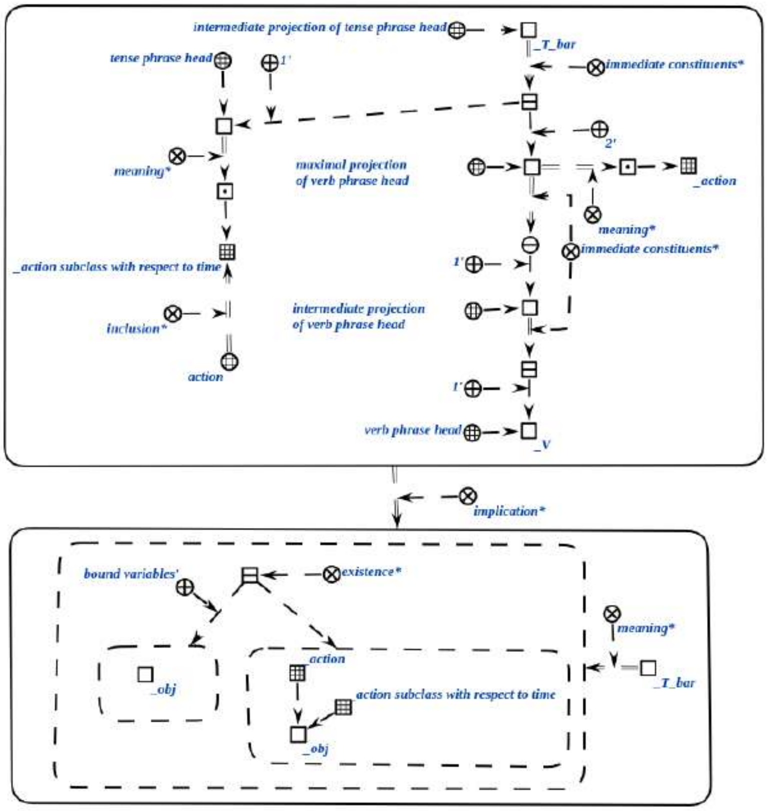


Figure 8. Semantic interpretation rule for an intermediate projection of a tense phrase head.

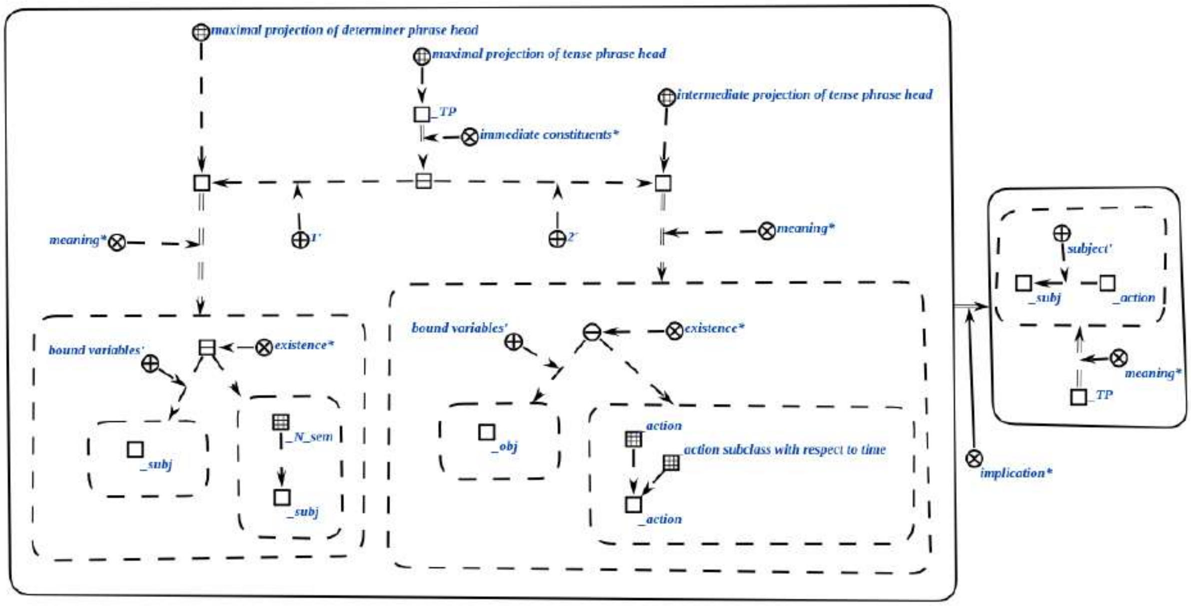


Figure 9. Semantic interpretation rule for a maximal projection of a tense phrase head (with an intransitive verb).

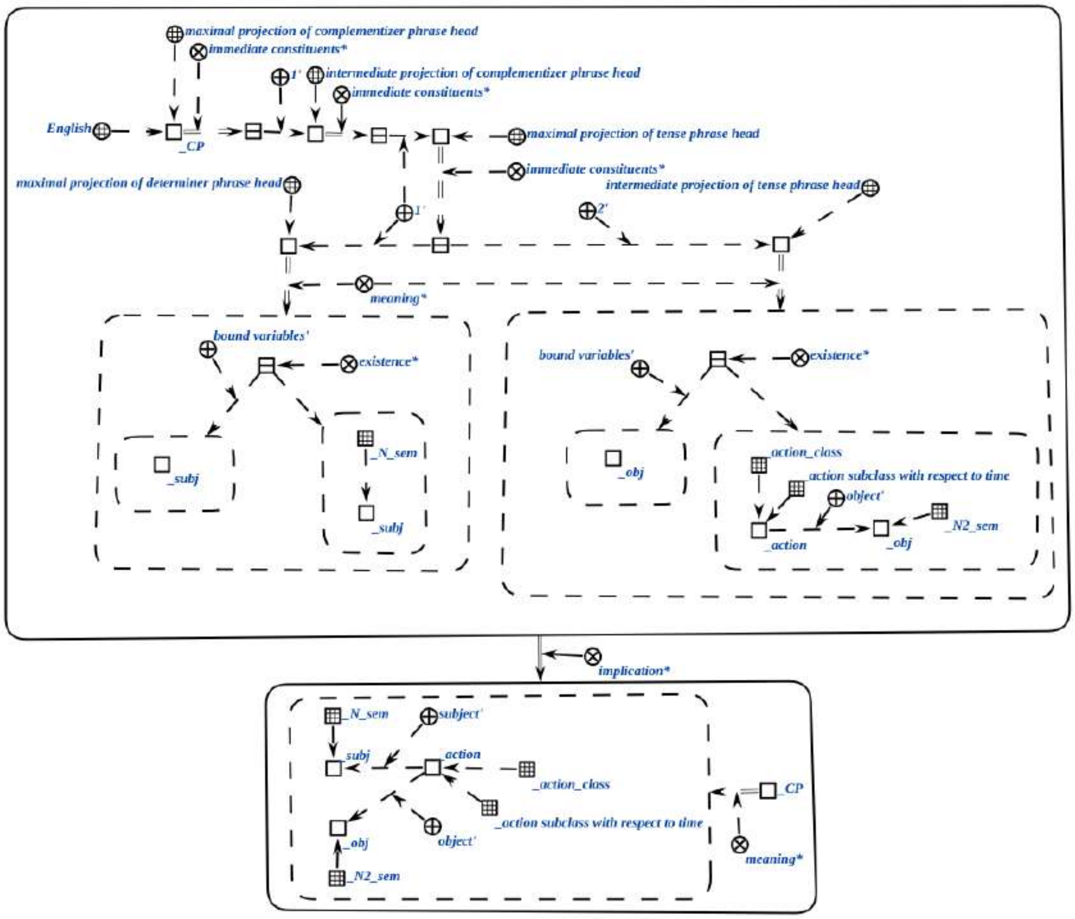

maximal projection of complementizer phrase head
immediate constituents*
1'
intermediate projection of complementizer phrase head
immediate constituents*
English
_CP
maximal projection of tense phrase head
immediate constituents*
maximal projection of determiner phrase head
intermediate projection of tense phrase head
1'
2'
meaning*
bound variables'
existence*
_subj
_N_sem
_subj
bound variables'
existence*
_obj
action_class
action subclass with respect to time
object'
_action
_obj
_N2_sem
implication*
_N_sem
subject'
subj
action
_action_class
_action subclass with respect to time
object'
_obj
_N2_sem
_CP
meaning*

Figure 10. Semantic interpretation rule for sentences with transitive verbs

reduce the overhead in the development of various systems that use them.

As a result a fragment of the upper-level ontology has been obtained, using which it becomes possible to formalize subject domains of specific languages, both natural and artificial, for their further use in natural-language interfaces of next-generation intelligent computer systems.

The fragment of ontology provided above is by no means complete. Given that, one of the directions of further study on this topic is extending the ontology, which would include at least the following:

- creating an ontology of lexical meanings in natural languages;
- describing the mechanism of matching tokens with lexemes in the knowledge base.

To add the ability to process a new language by a system designed on the basis of the approach proposed in this paper, it is necessary to create an ontology of this language, based on the presented upper-level ontology, which would include the following:

- vocabulary with lexemes of that language;
- specific features of syntax and vocabulary for that language;
- features of semantic interpretation specific to that language.

Acknowledgment

The authors would like to thank the research teams of the Department of Intellectual Information Technologies of BSUIR and the Department of Theory and Practice of Translation #1 of MSLU for their help and valuable comments. This research was partially supported by the BRFFR (BRFFR-RFFR No. F21RM-139).

References

[1] Golenkov, V. V., "Methodological problems of the current state of works in the field of artificial intelligence," *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, pp. 17–24, 2021.

[2] S. Pileggi, A. Lopez-Lorca, and G. Beydoun, "Ontologies in software engineering," 11 2018.

[3] P. Lando, A. Lapujade, G. Kassel, and F. Fürst, "Towards a general ontology of computer programs." 01 2007, pp. 163–170.

[4] S. Farrar, W. D. Lewis, and T. Langendoen, "A common ontology for linguistic concepts," 2002.

[5] C. Chiarcos, *Interoperability of Corpora and Annotations*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 161–179. [Online]. Available: https://doi.org/10.1007/978-3-642-28249-2_16

[6] "Text Encoding Initiative," Available at: https://tei-c.org/, (accessed 2022, October).

[7] "EAGLES Recommendations for the Morphosyntactic Annotation of Corpora," Available at: https://home.uni-leipzig.de/burr/Verb/htm/LinkedDocuments/annotate.pdf, (accessed 2022, October).

[8] N. Ide and J. Pustejovsky, "What does interoperability mean , anyway ? toward an operational definition of interoperability for language technology," 2010.

[9] A. C. Schalley, "Ontologies and ontological methods in linguistics," *Language and Linguistics Compass*, vol. 13, no. 11, p. e12356, 2019, e12356 LNCO-0634.R3. [Online]. Available: https://compass.onlinelibrary.wiley.com/doi/abs/10.1111/lnc3.12356

[10] J. P. McCrae, P. Labropoulou, J. Gracia, M. Villegas, V. Rodríguez-Doncel, and P. Cimiano, "One ontology to bind them all: The meta-share owl ontology for the interoperability of linguistic datasets on the web," in *The Semantic Web: ESWC 2015 Satellite Events*, F. Gandon, C. Guéret, S. Villata, J. Breslin, C. Faron-Zucker, and A. Zimmermann, Eds. Cham: Springer International Publishing, 2015, pp. 271–282.

[11] "The General Ontology of Linguistic Description," Available at: http://linguistics-ontology.org/info/about, (accessed 2022, October).

[12] A. Pease, I. Niles, and J. Li, "The suggested upper merged ontology: A large ontology for the semantic web and its applications," 01 2002.

[13] S. Farrar and D. Langendoen, "A linguistic ontology for the semantic web," *Glot International*, vol. 7, pp. 97–100, 03 2003.

[14] C. Chiarcos, "Ontologies of linguistic annotation: Survey and perspectives," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 303–310. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2012/pdf/911_Paper.pdf

[15] "The Theoretical Status of Ontologies in Natural Language Processing," Available at: https://arxiv.org/abs/cmp-lg/9704010, (accessed 2022, October).

[16] J. A. Bateman and G. Fabris, "The generalized upper model knowledge base: Organization and use," 2002.

[17] P. Buitelaar, P. Cimiano, P. Haase, and M. Sintek, "Towards linguistically grounded ontologies," in *The Semantic Web: Research and Applications*, L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. Simperl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 111–125.

[18] T. Kostareva, S. Chuprina, and A. Nam, "Using ontology-driven methods to develop frameworks for tackling nlp problems," in *AIST*, 2016.

[19] O. Nevzorova and V. Nevzorov, "Ontology-driven processing of unstructured text," in *Artificial Intelligence*, S. O. Kuznetsov and A. I. Panov, Eds. Cham: Springer International Publishing, 2019, pp. 129–142.

[20] P. Cimiano, J. Lüker, D. Nagel, and C. Unger, "Exploiting ontology lexica for generating natural language texts from RDF data," in *Proceedings of the 14th European Workshop on Natural Language Generation*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 10–19. [Online]. Available: https://aclanthology.org/W13-2102

[21] N. Bouayad-Agha, G. Casamayor, and L. Wanner, "Natural language generation in the context of the semantic web," *Semantic Web*, vol. 5, pp. 493–513, 01 2014.

[22] D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, and F. Özcan, "Athena: An ontology-driven system for natural language querying over relational data stores," *Proc. VLDB Endow.*, vol. 9, no. 12, p. 1209–1220, aug 2016. [Online]. Available: https://doi.org/10.14778/2994509.2994536

[23] M. Shamsfard and A. A. Barforoush, "Learning ontologies from natural language texts," *International Journal of Human-Computer Studies*, vol. 60, no. 1, pp. 17–63, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1071581903001368

[24] J. A. Bateman, J. Hois, R. Ross, and T. Tenbrink, "A linguistic ontology of space for natural language processing," *Artificial Intelligence*, vol. 174, no. 14, pp. 1027–1071, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370210000858

[25] M. Moens and M. Steedman, "Temporal ontology in natural language," in *Proceedings of the 25th Annual Meeting on Association for Computational Linguistics*, ser. ACL '87. USA: Association for Computational Linguistics, 1987, p. 1–7. [Online]. Available: https://doi.org/10.3115/981175.981176

[26] A. Dobrov, A. Dobrova, P. Grokhovskiy, M. Smirnova, and N. Soms, "Computer ontology of tibetan for morphosyntactic disambiguation," in *Digital Transformation and Global Society*, D. A. Alexandrov, A. V. Boukhanovsky, A. V. Chugunov, Y. Kabanov, and O. Koltsova, Eds. Cham: Springer International Publishing, 2018, pp. 336–349.

[27] "WordNet: A Lexical Database for English," Available at: https://wordnet.princeton.edu, (accessed 2022, October).

[28] "VerbNet: A Computational Lexical Resource for Verbs," Available at: https://verbs.colorado.edu/verbnet/, (accessed 2022, October).

[29] "FrameNet," Available at: http://framenet.icsi.berkeley.edu, (accessed 2022, October).

[30] A. Pease and C. Fellbaum, *Formal ontology as interlingua: the SUMO and WordNet linking project and global WordNet*, ser. Studies in Natural Language Processing. Cambridge University Press, 2010, p. 25–35.

[31] T. Matsukawa and E. Yokota, "Development of the concept dictionary implementation of lexical knowledge," in *Lexical Semantics and Knowledge Representation*, 1991. [Online]. Available: https://aclanthology.org/W91-0219

[32] N. Calzolari, "Acquiring and representing semantic information in a lexical knowledge base." 06 1991, pp. 235–243.

[33] P. Buitelaar, T. Declerck, A. Frank, S. Racioppa, M. Kiesel, M. Sintek, R. Engel, M. Romanelli, D. Sonntag, B. Loos, V. Micelli, R. Porzel, and P. Cimiano, "LingInfo: Design and Applications of a Model for the Integration of Linguistic Information in Ontologies," 2006. [Online]. Available: http://smartweb.dfki.de/Vortraege/OntoLex2006.pdf

[34] P. Cimiano, P. Haase, M. Herold, M. Mantel, and P. Buitelaar, "Lexonto: A model for ontology lexicons for ontology-based nlp," 2007.

[35] P. Buitelaar, M. Sintek, and M. Kiesel, "A multilingual/multimedia lexicon model for ontologies," in *The Semantic Web: Research and Applications*, Y. Sure and J. Domingue, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 502–513.

[36] J. McCrae, G. Cea, P. Buitelaar, P. Cimiano, T. Declerck, A. Gomez-Perez, J. Gracia, L. Hollink, E. Montiel-Ponsoda, D. Spohr, and T. Wunner, "Interchanging lexical resources on the semantic web," *Language Resources and Evaluation*, vol. 46, pp. 701–719, 12 2012.

[37] ""semantic web": W3c's vision of the web of linked data." Available at: https://www.w3.org/standards/semanticweb/, (accessed 2022, October).

[38] "RDF: a standard model for data interchange on the Web." Available at: https://www.w3.org/RDF/, (accessed 2022, October).

[39] "The Resource Description Framework. Abstract syntax (a data model) which serves to link all RDF-based languages and specifications." Available at: https://www.w3.org/TR/rdf11-concepts/, (accessed 2022, October).

[40] "NLTK NLP library," Available at: https://www.nltk.org/, (accessed 2022, October).

[41] "spaCy NLP library," Available at: https://spacy.io/, (accessed 2022, October).

[42] T. N. Erekhinskaya, M. Tatu, M. Balakrishna, S. Patel, D. Strebkov, and D. I. Moldovan, "Ten ways of leveraging ontologies for rapid natural language processing customization for multiple use cases in disjoint domains," *Open J. Semantic Web*, vol. 7, pp. 33–51, 2020.

[43] V. V. Golenkov, N. A. Gulyakina, D. V. Shunkevich, *Open technology for ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, G. V.V., Ed. Minsk: Bestprint, 2021.

[44] "SIL: Glossary of Linguistic Terms," Available at: https://glossary.sil.org, (accessed 2022, October).

[45] D. Adger, *Core Syntax: A Minimalist Approach*, ser. Core linguistics. Oxford University Press, 2003. [Online]. Available: https://books.google.by/books?id=GMJ1QgAACAAJ

[46] R. Jackendoff and R. Jackendoff, *X Syntax: A Study of Phrase Structure*, ser. Linguistic inquiry monographs. MIT Press, 1977. [Online]. Available: https://books.google.by/books?id=ALf6PgAACAAJ

[47] L. Haegeman, *Introduction to Government and Binding Theory*, ser. Blackwell Textbooks in Linguistics. Wiley, 1994. [Online]. Available: https://books.google.by/books?id=_fvUInHLUTwC

[48] A. Carnie, *Syntax: A Generative Introduction*, ser. Introducing Linguistics. Wiley, 2012. [Online]. Available: https://books.google.by/books?id=MFZ1UV3YGtgC

[49] I. Heim and A. Kratzer, *Semantics in Generative Grammar*, ser. Blackwell Textbooks in Linguistics. Wiley, 1998. [Online]. Available: https://books.google.by/books?id=jAvR2DB3pPIC

[50] Y. Winter, *Elements of Formal Semantics*. Edinburgh: Edinburgh University Press, 2016. [Online]. Available: https://doi.org/10.1515/9780748677771

[51] P. Portner and B. Partee, *Formal Semantics: The Essential Readings*, ser. Linguistics: The Essential Readings. Wiley, 2008. [Online]. Available: https://books.google.by/books?id=ptgUWREtAkMC

Средства формального описания синтаксиса и денотационной семантики различных языков в интеллектуальных компьютерных системах нового поколения

Гойло А. А., Никифоров С. А.

Статья посвящена языковым средствам формального описания синтаксиса и денотационной семантики различных языков в интеллектуальных компьютерных системах нового поколения. Предложена формальная онтология различных языков. Уточнена трактовка таких понятий как язык, знак, информационная конструкция, знаковая конструкция, синтаксис, семантика, смысл, значение и др. Формализованы предметные области синтаксиса и денотационной семантики естественных языков. В результате получена онтология верхнего уровня, с использованием которой становится возможной формализация более частных предметных областей конкретных языков как естественных, так и искусственных для их дальнейшего применения в естественно-языковых интерфейсах интеллектуальных систем нового поколения.

Received 28.10.2022

Hybrid problem solvers of intelligent computer systems of a new generation

Daniil Shunkevich
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: shunkevich@bsuir.by

***Abstract*—In the article, the actual problems of the current state of technologies for the development of hybrid problem solvers are formulated, an approach to their solution based on the OSTIS Technology is proposed. The principles of building a problem solver as a hierarchical system of skills based on a multi-agent approach are formulated, the ontologies of agents and the actions performed by them are given. The principles of synchronization of agents' activities are formulated, as well as the ontology of the basic programming language for implementing agent programs and the interpreter model of such a language are developed.**

***Keywords*—OSTIS, problem solver, multi-agent system, problem-solving model, ontological approach**

I. Introduction

Currently, the usage of intelligent systems in a variety of fields is becoming increasingly relevant. One of the key components of an intelligent system that provides the ability to solve a wide range of problems is a problem solver. Its peculiarity in comparison with other modern software systems is the need to solve problems in conditions when the necessary information is not explicitly localized in the knowledge base of the intelligent system and must be found in the process of solving the problem, based on any criteria.

In other words, if in traditional systems, when solving a problem, it is always assumed that there are some localized source data ("given") and some description of the desired result ("what is required"), then in an intelligent system, all the information currently available in the system acts as source data when solving a large number of problems, that is, the entire knowledge base. In addition, if it is impossible to solve the problem in the current state of the knowledge base, the intelligent system should be able to understand what exactly is missing to continue the solution process and try to get the missing information in the external environment (for example, to request from the user).

To date, within various fields of artificial intelligence, a large number of different *problem-solving models* have been developed, each of which allows solving problems of a certain class. The expansion of the application fields for intelligent systems requires them to be able to solve so-called complex problems, the solution of each of which requires combining several problem-solving models, while it is not known a priori in what order and how many times one or another model will be used. Problem solvers, in which several problem-solving models are combined, are called *hybrid problem solvers*, and intelligent systems, in which various types of knowledge and various problem-solving models are combined, are called *hybrid intelligent systems* [1].

Improving the efficiency of the development and maintenance of hybrid intelligent systems requires the unification of models for the representation of various knowledge types and knowledge processing models, which would simplify the integration on its basis of components corresponding to different problem-solving models. Such models based on a unified semantic representation of information are proposed within an Open semantic technology for intelligent systems design (*OSTIS Technology*) [2] and are described within the corresponding *OSTIS Standard* [3]. In the article [2], the analysis of modern approaches to the development of hybrid problem solvers is carried out and it is shown that the approach proposed within the *OSTIS Technology* is currently the only example of a comprehensive approach to the development of hybrid problem solvers, within which the above problems are solved.

However, there are a number of problems that remain relevant and require solutions.

II. Current problems in the development of hybrid problem solvers

The first problem is related to the lack of a sufficiently strict formalized classification of problems solved by intelligent systems, the lack of unification in the description of problems and classes of problems, the description of purposes, progress, and result of solving the problem, problem-solving methods, relations between classes of problems and problem-solving methods of this class. The solution of this problem, on the one hand, will allow for the possibility of deep integration of various problem-solving models of various classes and the ability to simplify the process of integrating new problem-solving models into an intelligent system and, on the other hand, will become a precondition for solving other problems described below.

The second problem is that at the moment the main attention in the field of developing hybrid problem solvers is paid to reducing the complexity of integrating various components of the problem solver into an intelligent system and realizing the possibility of accumulating reusable solvers components, but in general it is not said how specifically the intelligent system will use certain components in solving problems of specific classes. Thus, the creation of a general plan for solving a problem, i.e. the selection of problem-solving methods, the determination of the order of their application, and the choice of source data (arguments) for the usage of a particular method is actually determined by the developer at the stage of system design or its evolution during operation. The precondition for solving this problem is the solution of the previously considered problem of unifiying the representation of problems of various classes and methods for solving them. The solution of the problem under consideration involves the development of a set of *problem-solving strategies* (or problem-solving meta-methods) that will allow the intelligent system to independently form a plan for solving the problem, taking into account the problem-solving methods available in the system and, if possible, even request the missing components for solving the problem in the appropriate libraries. It should be noted that attempts to develop universal high-level approaches to solving problems were made at the dawn of the development of artificial intelligence, in the 1950s and 60s, but were unsuccessful and soon be abandoned. This is largely conditioned by the lack of unified models of knowledge representation and processing at that time, which are currently proposed within the *OSTIS Technology*.

Another urgent problem, closely related to those discussed above, is that intelligent systems are often forced to solve problems in the conditions of so-called non-factors, that is, when the description of the problem and possible ways to solve it are incomplete, the fuzziness and incorrectness of existing knowledge, as well as the lack of criteria for evaluating the optimality of the resulting solution, etc. take place [4]. This is especially relevant when solving behavioral problems related to changes in the state of objects of the environment external to the intelligent system. To solve problems in such conditions, an intelligent system must not only have a sufficient set of problem solver components that implement problem-solving models in the presence of non-factors (fuzzy logic models, machine learning models, genetic algorithms, etc.) but also implement *problem-solving strategies* that would allow making decisions and forming a plan for solving the problem in such conditions.

Problems considered are primarily related to the process of solving a specific problem by an intelligent system. At the same time, it is obvious that at every moment of time, an intelligent system is forced to solve several problems in parallel, which can be related both to the direct functional purpose of the system and to ensuring the operation and evolution of the system itself. In the second case, the problems related to updating the information it contains about the outside world, finding and eliminating errors in the knowledge base, optimizing the structure of the knowledge base and the solver of the system, finding and eliminating information garbage, and many others are meant. At the same time, different problems may have different priorities, which may vary depending on the situation, even in the process of its solution. At the same time, in a situation where it is not known a priori which of the possible ways to solve the problem will be the most effective, it may be advisable to use several approaches in parallel to solve the same problem. Thus, the problem of organizing the control of information processes for solving problems in an intelligent system and the interaction of information processes that occur in parallel, taking into account the priority of processes, the ability to monitor the current state of information processes, generate, suspend, and eliminate information processes is relevant. To solve this problem, it is advisable to borrow solutions widely used in traditional computer systems, in particular, implemented in modern operation systems, and adapt them to the specifics of solving problems in intelligent systems. It is important to note that the implementation of the information process control model, based on the general unified information processing models proposed within the OSTIS Technology, will make some information processes the object of analysis of other information processes, which, in turn, will make it possible to analyze the progress of solving the problem directly in the process of solving, evaluate the effectiveness of certain problem-solving methods, collect the most successful solutions for its further application in solution of the similar problems, and much more.

Solving these problems will allow developing a fundamentally new hierarchical model of a *hybrid problem solver*, which has a number of significant advantages, which, in turn, will have to be interpreted on any platforms. Without unifying the requirements for the platform of interpreting intelligent systems models and a clear separation of the platform-independent model of the system (and, in particular, the solver) and the platform, it is impossible to talk about the implementation of the solver model realizing the ideas discussed above. This will lead to the need to duplicate the same model components for different platforms and will significantly complicate the integration of solver components, since it will require taking into account the features of each platform during such integration. In addition, a clear separation of the system model level and the platform level will make it possible to independently develop various platforms and models of intelligent systems. Thus, it is proposed to formulate unified requirements for the platform of interpreting semantic models of intelligent systems, as well as to build a general model of such a platform that meets these requirements.

On the other hand, as already mentioned, the problem solver is a complex system focused on working with knowledge, not with data, unlike modern software systems in which it is initially known where exactly the necessary data is localized and in what form they are represented. In this regard, the usage of modern hardware and software platforms, focused on address access to data stored in memory, for the development of intelligent systems is not always effective, since when developing intelligent systems, it is actually necessary to model nonlinear memory based on linear one. Increasing the efficiency of problem solving by intelligent systems requires the development of specialized platforms, including hardware ones, focused on unified semantic models of information representation and processing. As a basis for such developments, it is proposed to use the suggested within the *OSTIS Technology* general concepts of a semantic computer, semantic memory, and a basic programming language focused on processing information in such memory, and complement them with ideas of wave programming languages, insertion programming, and other approaches aimed at improving the efficiency of knowledge processing, including at the hardware level.

The development of problem solvers, including the problems of developing hybrid problem solvers discussed above, are currently being considered in the context of single (independent) intelligent systems operating in some environment (of which the user is also a part, if there is one). At the same time, there is an obvious tendency of modern information technologies to move from single systems to collectives of distributed interacting computer systems, in particular, to distributed data storage and distributed computing. In the case of intelligent computer systems, as the most important property of the systems included in such collectives, *interoperability* serves, that is, the ability of the system to coordinate interaction with other similar systems in order to solve any problems. Thus, the transition from the development of problem solvers of individual intelligent systems to problem solvers of interacting interoperable intelligent systems is particularly relevant, including the development of principles for solving problems in such distributed collectives, taking into account the solution of all the problems outlined above. To solve this problem, it is proposed to apply the ideas suggested within the theory of multi-agent systems and reinterpreted in the context of the interaction of hybrid intelligent systems.

In addition, the most important problem in the case of a distributed collective of intelligent systems is not just providing the ability to solve problems by such a collective at the current time but permanently supporting semantic compatibility and, as a consequence, the interoperability of systems included in such a collective throughout their entire life cycle. It is obvious that each of the systems included in such a collective and, accordingly, its problem solver can evolve independently of other systems, but at the same time, interoperability between systems must always be maintained, otherwise solving problems in such a collective will become impossible. The solution of this problem involves the development of methods for permanently analyzing semantic compatibility of a distributed collective of interacting intelligent systems, identification and elimination of problems.

Within this article, an approach to solving some of the listed problems based on the *OSTIS Technology* is proposed.

III. Proposed approach

As mentioned earlier, it is proposed to solve these problems within the *OSTIS Technology*. Let us list the basic principles of this technology that create preconditions for solving these problems:

- The *OSTIS Technology* is based on a universal method of semantic representation (encoding) of information in the memory of intelligent computer systems, called an *SC-code*. Texts of the *SC-code* (sc-texts, sc-constructions) are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, depending on orientation, can be *sc-arcs* or *sc-edges*). The *Alphabet of the SC-code* consists of five main elements, on the basis of which SC-code constructions of any complexity are built, including more specific types of sc-elements (for example, new concepts). The universality and uniformity of the *SC-code* makes it possible to describe on its basis any *knowledge types* and any problem-solving *methods*, which, in turn, greatly simplifies their integration within one system. Systems developed based on the *OSTIS Technology* are called *ostis-systems*;
- The basis of the knowledge base developed by the *OSTIS Technology* is a hierarchical system of semantic models of *subject domains* and *ontologies*, among which the universal *Kernel of the knowledge base semantic models* and the methodology for the development of semantic knowledge base models are allocated, which ensure the semantic compatibility of the knowledge bases being developed;
- The basis of information processing within the *OSTIS Technology* is the *SCP Language*, the program texts of which are also written in the form of SC-code constructions;

- The problem solver architecture within the *OSTIS Technology* is based on a multi-agent approach, in which agents interact with each other purely by specifying the actions they perform within a common semantic memory (such agents are called *sc-agents*). Such an approach allows ensuring the fundamental possibility of implementing any *problem-solving methods* in the form of corresponding solver components and providing their semantic compatibility. Other advantages of the multi-agent approach in general are widely known and discussed in related publications [5]–[7].

The listed principles of the *OSTIS Techology* are proposed to be supplemented with some of the ideas underlying the solution of those problems and, taking this into account, to develop:

- A complex ontology of actions, problems, and methods of their solution, as well as an ontology of *hybrid problem solvers*, on the basis of which to clarify the concept of the solver and its architecture. The first version of the *Global subject domain of actions and problems and the corresponding ontology of methods and technologies* is already represented within the *OSTIS Standard*, on its basis it is proposed to develop an ontology of actions and problems solved by *ostis-systems*;
- A complex of unified generalized strategies (meta-methods) for solving problems in intelligent systems, which allows an intelligent system to independently form a plan for solving a problem, taking into account the problem-solving methods available in the system. In addition to the experience of similar works, it is also proposed to supplement the developed strategies with some general methodological ideas related to the theory of behaviorism and the ideas of its application in computer science that are gaining popularity [8]–[10], TIPS [11], as well as the STA-methodology proposed by the school of G. Shchedrovitsky [12];
- An ontological model for the formation of a plan for solving a problem and managing the process of solving problems in hybrid problem solvers under conditions of various non-factors and the absence of clear criteria for evaluating the optimality of the resulting solution. To develop this model, it is proposed to adapt the theory of situational control proposed by D. Pospelov [13] and implement it in the context of the semantic theory of problem solvers developed within the OSTIS Technology;
- An ontological model for controlling information processes for solving problems in intelligent systems built on the basis of unified semantic models of information representation and processing;
- An ontological model of the platform for interpreting unified semantic models of information representation and processing (*ostis-platforms*);
- A comprehensive hierarchical model of a hybrid problem solver based on a multi-agent approach and taking into account the need to solve problems both within single intelligent systems and within distributed collectives of interoperable intelligent systems;
- A complex of methods for analyzing the quality of hybrid problem solvers and their components;
- A complex of tools to support the design of hybrid problem solvers.

Within the *OSTIS Technology*, several universal variants of visualization of *SC-code* constructions are proposed, such as *SCg-code* (graphic variant), *SCn-code* (nonlinear hypertext variant), *SCs-code* (linear string variant). Within this article, fragments of structured texts in the SCn code [3] will often be used, which are simultaneously fragments of the source texts of the knowledge base, understandable to both human and machine. This allows making the text more structured and formalized, while maintaining its readability. The symbol ":=" in such texts indicates alternative (synonymous) names of the described entity, revealing in more detail certain of its features.

As follows from the principles of the *OSTIS Technology* discussed earlier, the building of ontological models of any entities involves the development of an appropriate *subject domain and ontology* (or a family of *subject domains and ontologies*), within which the properties of this entity are clarified by a formal description of the corresponding set of concepts, including relations.

Within this work, we will consider in more detail the fragments of:

- The *Subject domain and ontology of ostis-systems problem solvers*, which clarifies the concepts of a problem solver, a knowledge processing machine, as well as the classification of problem solvers and knowledge processing machines;
- The *Subject domain and ontology of actions and problems of ostis-systems*, within which the classes of actions and problems solved in ostis-systems are specified;
- The *Subject domain and ontology of sc-agents*, which clarifies the concept of an sc-agent as a component of the ostis-system problem solver, the typology of sc-agents and their properties, as well as the principles for synchronizing the activities of sc-agents;
- The *Subject domain and ontology of the Basic programming language of ostis-systems*, which clarifies the syntax, denotational semantics, and operational semantics of the SCP Language, which is the basic for ostis-systems.

IV. Ostis-systems problem solvers

Within the *OSTIS Technology*, the *ostis-system problem solver* is defined as the totality of all *skills* possessed by the ostis-system at the current time [3], [14].

In turn, the skill is interpreted as a combination of some *method* and its operational semantics, that is, information about how this *method* should be interpreted.

By a *method* we will understand the description of how any or almost any (with explicit exceptions) action belonging to the corresponding *action class* can be performed. Since a specific *action class* corresponds to some specific *problem class*, we can say that the method describes a way to solve any problems belonging to a given class. The concept of a method can be considered as a generalization of the concept of "program", in connection with which, within the OSTIS Technology, the terms "method" and "program" are synonymous [14].

As an example of a particular method, a procedural program in a specific programming language or a set of logical propositions that make up a formal theory of a given subject domain (analogous to a logical program) can be used.

A particular case of the method is the program of the atomic component of the *ostis-system problem solver* (*atomic sc-agent*); in this case, a collective of lower-level agents, interpreting the corresponding program, acts as the operational semantics of the method (in the extreme case, these will be agents that are part of the platform for interpreting computer system models, including the hardware one).

Thus, we can talk about the hierarchy of *methods* and *methods* for interpreting other *methods*. Taking into account this thesis, it is possible to clarify the concept of a problem solver as a hierarchical system of skills.

An approach to the building of problem solvers proposed within the *OSTIS Technology* allows them to be modifiable, which, in turn, allows the *ostis-system*, if necessary, to easily acquire new *skills*, modify (improve) existing ones, and even get rid of some skills in order to improve system performance. Thus, it makes sense to talk not about a rigidly fixed problem solver, which is developed once when creating the first version of the system and does not change further, but about a set of skills fixed at each current moment of time but constantly evolving.

ostis-system problem solver
⇐ *family of subsets**:
  skill
:= [hierarchical system of skills possessed by the ostis-system]
⊃ *hybrid ostis-system problem solver*
  := [ostis-system problem solver that implements two or more problem-solving models]
⊃ *combined ostis-system problem solver*
  := [complete ostis-system problem solver]
  := [integrated ostis-system problem solver]
  := [ostis-system problem solver that implements all its functionality, both basic and auxiliary]

In general, the *combined ostis-system problem solver* solves problems related to:

- providing the basic functionality of the system (for example, solving explicitly formulated problems at the user's request);
- ensuring the correctness and optimization of the ostis-system itself (permanently throughout the entire life cycle of the ostis-system);
- providing advanced training for end users and developers of the ostis-system;
- providing automation of the design and control of the development of the ostis-system.

By a *knowledge processing machine* we will understand the set of interpreters of all *skills* that make up some *problem solver*. Taking into account the multi-agent approach to information processing used within the OSTIS Technology, the *knowledge processing machine* is an *sc-agent* (most often – a *non-atomic sc-agent*), which includes simpler sc-agents that provide interpretation of the corresponding set of *methods*. Thus, the *knowledge processing machine* in general is a hierarchical system of *sc-agents*.

Taking into account the fact that there is a hierarchy of methods in terms of the level of interpretation (some methods interpret others), it is also necessary to talk about the hierarchy of *knowledge processing machines*.

knowledge processing machine
⊂ *sc-agent*

Let us consider the classification of ostis-systems problem solvers according to various criteria.

Classification of ostis-systems problem solvers by the type of the corresponding ostis-system:

ostis-system problem solver
∋ *Problem solver of the IMS.ostis Metasystem*
⊃ *problem solver of the auxiliary ostis-system*
  ⊃ *problem solver of the computer system interface*
  ⊃ *ostis-subsystem problem solver for supporting the design of components of a certain class*
    ⊃ *ostis-subsystem problem solver for supporting knowledge base design*
    ⊃ *ostis-subsystem problem solver for supporting the design of ostis-systems problem solvers*
  ⊃ *problem solver of the control subsystem for the design of computer systems and their components*

⊃ *problem solver of an independent ostis-system*

problem solver of the computer system interface
⇒ *subdividing**:
{● *problem solver of the user interface of a computer system*
● *problem solver of the computer system interface with other computer systems*
● *problem solver of the computer system interface with the environment*
}

ostis-subsystem problem solver for supporting knowledge base design
⊃ *problem solver for improving the quality of the knowledge base*
  ⊃ *problem solver for knowledge base verification*
    ⊃ *problem solver for finding and eliminating inaccuracies in the knowledge base*
    ⊃ *problem solver for finding and eliminating incompleteness*
  ⊃ *problem solver for optimizing the structure of the knowledge base*
  ⊃ *problem solver for identifying and eliminating information garbage*

ostis-subsystem problem solver for supporting the design of ostis-systems problem solvers
⇒ *subdividing**:
{● *ostis-subsystem problem solver for supporting the design of knowledge processing programs*
● *ostis-subsystem problem solver for supporting the design of knowledge processing agents*
}

Classification of ostis-systems problem solvers by the type of interpreted problem-solving model:

ostis-system problem solver
⊃ *problem solver with stored methods*
  := [solver capable of solving problems of those classes for which the corresponding solution method is known at a given moment]
  ⊃ *problem solver based on neural network models*
  ⊃ *problem solver based on genetic algorithms*
  ⊃ *problem solver based on imperative programs*
    ⊃ *problem solver based on procedural programs*
    ⊃ *problem solver based on object-oriented programs*
  ⊃ *problem solver based on declarative programs*
    ⊃ *problem solver based on logical programs*
    ⊃ *problem solver based on functional programs*
⊃ *problem solver in conditions when the method of solving problems of this class is not known at the current time*
  := [solver that implements problem-solving strategies that allow generating a problem-solving method that is not currently known to the ostis-system]
  := [solver that uses meta-methods for solving problems, corresponding to more general classes of problems in relation to a given one]
  := [problem solver that allows generating a method that is particular in relation to any method known to the ostis-system and is interpreted by the corresponding knowledge processing machine]
  ⊃ *solver that implements the strategy of finding ways to solve the problem in depth*
  ⊃ *solver that implements a strategy for finding ways to solve a problem in width*
  ⊃ *solver that implements a trial-and-error strategy*
  ⊃ *solver that implements a strategy for splitting a problem into subproblems*
  ⊃ *solver that implements a strategy for solving problems by analogy*
  ⊃ *solver that implements a concept of an intelligent software package*

Separately, we will highlight the classification of knowledge processing machines, which in general can correspond to the same fragments of the knowledge base but together with them form different skills and, accordingly, different problem solvers:

knowledge processing machine
⊃ *logical inference machine*
  ⊃ *deductive inference machine*
    ⊃ *direct deductive inference machine*
    ⊃ *reverse deductive inference machine*
  ⊃ *inductive inference machine*
  ⊃ *abductive inference machine*
  ⊃ *fuzzy inference machine*
  ⊃ *inference machine based on default logic*
  ⊃ *logical inference machine with*

consideration for the time factor

Classification of ostis-systems problem solvers by the type of problem to be solved (purposes of solving the problem):

ostis-system problem solver

- ⊃ *problem solver for information search*
  - ⇒ *subdividing**:
    - {• *problem solver for finding information that meets the specified criteria*
    - • *problem solver for finding information that does not meet the specified criteria*
    - }
- ⊃ *solver of explicitly formulated problems*
  - := [problem solver for which the purpose is explicitly formulated]
  - ⊃ *problem solver for searching or calculating the values of a given set of quantities*
  - ⊃ *problem solver for establishing the truth of a given logical proposition within a given formal theory*
  - ⊃ *problem solver for forming a proof of a given proposition within a given formal theory*
  - ⊃ *machine for verifying the response to the specified problem*
  - ⊃ *machine for verifying the solution of the specified problem*
    - ⊃ *machine for verifying the proof of a given proposition within a given formal theory*
- ⊃ *problem solver for entity classification*
  - ⊃ *machine for correlating an entity with one of a given set of classes*
  - ⊃ *machine for dividing a set of entities into classes according to a given set of attributes*
- ⊃ *problem solver for the synthesis of information constructions*
  - ⊃ *problem solver for the synthesis of natural language texts*
  - ⊃ *problem solver for image synthesis*
  - ⊃ *problem solver for signal synthesis*
    - ⊃ *problem solver for speech synthesis*
- ⊃ *problem solver for the analysis of information constructions*
  - ⊃ *problem solver for analysis of natural language texts*
    - ⊃ *problem solver for understanding natural language texts*
    - ⊃ *problem solver for verification of natural language texts*
  - ⊃ *problem solver for image analysis*
    - ⊃ *problem solver for image segmentation*
    - ⊃ *problem solver for understanding images*
  - ⊃ *problem solver for signal analysis*
    - ⊃ *problem solver for speech analysis*
      - ⊃ *problem solver of speech understanding*

V. General principles of information processing in ostis-systems

The proposed approach to problem solving is based on a number of ideas related to the concept of situational control proposed in the work of D. Pospelov [13]. To date, attempts to implement this concept, despite its relevance and demand, have been reduced to particular solutions for specific classes of problems and, unfortunately, have not been widely distributed. To a large extent, this is conditioned by the lack of a universal unified basis that would make it possible to create situational control languages based on it in application to specific subject domains and, more importantly, reuse fragments of descriptions in such languages.

This problem can be solved using an *SC-code*, proposed within the *OSTIS Technology*, and a family of top-level ontologies developed on its basis. In particular, the implementation of the ideas of situational control is facilitated by such principles as:

- the SC-code as a basic language for describing any information in the knowledge base and, accordingly, for building situational control languages based on it;
- basic set-theoretic semantics of the SC-code, which makes it possible to formally clarify all the concepts used in the form of a formal set of ontologies, which allows for compatibility of the systems being developed and the possibility of reusage of their components;
- an agent-oriented approach to information processing, involving the reaction of agents to the occurrence of certain situations and events in the knowledge base.

Let us consider in more detail the basic principles of information processing underlying the proposed approach:

- The problem solver of each *ostis-system* is based on a multi-agent system whose agents interact with each other only(!) through their shared *sc-memory* by specifying in this memory the *actions in sc-memory* performed by them. At the same time, users of the *ostis-system* are also considered as agents of this system. In addition, *sc-agents* are divided into internal, receptor, and effector. Interaction between agents via shared *sc-memory* is reduced to the following types of actions:

1) usage of the part of the stored knowledge base that is available for the corresponding group of sc-agents;
2) formation (generation) of new fragments of the knowledge base and/or correction (editing) of any fragments of the available part of the knowledge base;
3) integration (immersion) of new and/or updated fragments into the available part of the knowledge base.

Let us emphasize that sc-agents do not communicate with each other directly by sending messages, as is done in most modern approaches to building multi-agent systems. In addition, sc-agents have access to a common knowledge base, which guarantees semantic compatibility (mutual understanding) between agents, including users of ostis-systems.

- The user of the *ostis-system* cannot directly perform any action in sc-memory, but via the user interface they can initiate the construction (generation, formation in *sc-memory*) of *sc-text*, which is a specification of the *action in sc-memory* performed either by one *atomic sc-agent* in one act, or by one *atomic sc-agent* in several acts, or by a collective of *sc-agents* (*non-atomic sc-agent*). In the specification of each such *action in sc-memory* initiated by a user, this user is indicated as the customer of this action. Thus, the user of the *ostis-system* gives instructions (tasks, commands) to *sc-agents* of this system to perform various actions specified by them in *sc-memory*.
- Each *sc-agent*, performing some *action in sc-memory*, have to "remember" that *sc-memory*, on which it is working, is a shared resource not only for it but also for all others *sc-agents*, working on the same *sc-memory*, therefore, the *sc-agent* must comply with a certain ethics for behaving in a collective of such *sc-agents*, which should minimize the interferences that it creates to other *sc-agents*.
- The activity of each agent of the *ostis-system* is discrete and represents a set of elementary actions (acts). At the same time, when performing each act, the agent can set several types of locks on fragments of the knowledge base. These locks allow prohibiting other agents from changing the specified fragment of the knowledge base or even making it "invisible" to other agents. The locks are set by the agent itself during the execution of the relevant act and are removed by it at the last stage of the execution of this act or earlier, if possible.
- If a certain *sc-agent* performs some *action in sc-memory*, then, for the duration of this action, it can:

1) prohibit other *sc-agents* from changing the state of some sc-elements stored in *sc-memory* – delete them, change the type;
2) prohibit other *sc-agents* from adding or deleting elements of some sets denoted by the corresponding *sc-nodes*;
3) prohibit other *sc-agents* from viewing some *sc-elements*, that is, these *sc-elements* become completely "invisible" (completely blocked) for other *sc-agents* but only for the duration of performing the proper action.

The specified locks must be completely removed before the completion of the corresponding action. Let us emphasize that the number of *sc-elements* blocked for the duration of some action mainly includes atomic and non-atomic connectives and should not include *sc-nodes* denoting infinite classes of any entities and, moreover, sc-nodes denoting various concepts (key classes of various subject domains).

Ethical (non-selfish) behavior of the *sc-agent* concerning blocking of *sc-elements* (that is, restricting access to them to other *sc-agents*) implies compliance with the following rules:

1) there should not be more *sc-elements* blocked than is necessary to solve the problem;
2) as soon as for any *sc-element* the need to lock it disappears before the completion of the corresponding action, it is advisable to immediately unlock this *sc-element* (remove the lock).

In order for the *sc-agent* to be able to work with any random *sc-element*, it must either make sure that this *sc-element* is not included in the knowledge base fragment that is part of the *full lock* or make sure that this lock is not set by this agent.

A special group of completely blocked *sc-elements* (for the duration of the action by the *sc-agent*) are auxiliary *sc-elements* ("scaffolds"), created only for the duration of this action. These sc-elements should not be unblocked at the end of the action but need to be deleted).

- If an *action in sc-memory* performed by the *sc-agent* has completed (i.e. has become a past entity), then the *sc-agent* registers the result of this *action*, specifying (1) deleted *sc-elements* and generated sc-elements. This is necessary if for some reason it will be required to rollback this *action*, i.e. to return to the state of the knowledge base before performing the specified *action*.

Let us list some advantages of the proposed approach to the organization of knowledge processing in ostis-systems:

- since processing is carried out by agents that exchange messages only through shared memory, adding a new agent or excluding (deactivating) one or more existing agents usually does not lead to

changes in other agents, since agents do not exchange messages directly;
- agent initiation is carried out in a decentralized manner and most often independently of each other, so even a significant expansion of the number of agents within one system does not lead to a deterioration in its performance;
- agent specifications and, as will be shown below, their programs can be written in the same language as the processed knowledge, which significantly reduces the list of specialized tools developed for the design of such agents and their collectives, as well as their analysis, verification, and optimization, and simplifies the development of the system by using more universal components.

VI. Actions and problems in ostis-systems

The building problem solvers and their components implies the need to describe the actions they perform and the problems they solve.

A. Concept of action in sc-memory

action in sc-memory
:= [internal action of the ostis-system]
:= [action performed in sc-memory]
:= [action performed in an abstract unified semantic memory]
:= [action performed by the ostis-system knowledge processing machine]
:= [action performed by an agent or a collective of agents of the ostis-system]
:= [information process on the knowledge base stored in sc-memory]
:= [process of solving an information problem in sc-memory]
⊂ *process in sc-memory*

Each ***action in sc-memory*** denotes some transformation performed by some *sc-agent* (or a collective of *sc-agents*) and focused on the transformation of *sc-memory*. The specification of the action after its execution can be included in the protocol for solving some problem.

The transformation of the state of the knowledge base includes, among other things, information search, which assumes (1) localization of the response to the request in the knowledge base, explicit allocation of the response structure, and (2) translation of the response into some external language.

The set of ***actions in sc-memory*** includes signs of actions of various kinds, the semantics of each of which depends on the specific context, i.e. the orientation of the action to any specific objects and the belonging of the action to any particular class of actions.

It should be clearly distinguished:
- each specific ***action in sc-memory***, which is some kind of transition process that transfers sc-memory from one state to another;
- each type of ***actions in sc-memory***, which is a certain class of similar actions (in one sense or another);
- sc-node denoting some specific ***action in sc-memory***;
- sc-node denoting a structure that is a description, specification, task, statement of the corresponding action.

Let us consider in more detail the classification of actions in sc-memory:

action in sc-memory
⊃ *action in sc-memory initiated by a question*
⊃ *action of editing the ostis-system knowledge base*
⊃ *action of setting the ostis-system mode*
⊃ *action of editing a file stored in sc-memory*
⊃ *action of interpreting a program stored in sc-memory*
  ⊃ *action of scp-program interpretation*

action in sc-memory initiated by a question
:= [action aimed at forming an answer to the question posed]
⊃ *action. create the specified file*
⊃ *action. create the specified structure*
  ⊃ *action. verify the specified structure*
    ⊃ *action. determine the truth or falsity of the indicated logical proposition*
    ⊃ *action. determine the correctness or incorrectness of the specified structure*
    ⊃ *action. create a structure describing the inaccuracies that exist in the specified structure*
  ⊃ *action. clarify the type of the specified sc-element*
    ⊃ *action. determine the positivity/negativity of the indicated sc-arc of belonging or non-belonging*
  ⊃ *action. create a semantic neighborhood*
    ⊃ *action. create a complete semantic neighborhood of the specified entity*
    ⊃ *action. create a basic semantic neighborhood of the specified entity*
    ⊃ *action. create a particular semantic neighborhood of the specified entity*
  ⊃ *action. create a structure describing the relations between the specified entities*
    ⊃ *action. create a structure*

describing the similarities of the specified entities

⊃ *action. create a structure describing the differences of the specified entities*

⊃ *action. create a structure describing the way to solve the specified problem*

⊃ *action. create a plan for generating an answer to the specified question*

⊃ *action. create a protocol for performing the specified action*

⊃ *action. create a justification for the correctness of the indicated solution*

⊃ *action. verify the justification of the correctness of the specified solution*

⊃ *action aimed at establishing the temporal characteristics of the specified entity*

⊃ *action aimed at establishing the spatial characteristics of the specified entity*

action of editing the knowledge base

⊃ *action. change the direction of the specified sc-arc*

⊃ *action. fix errors in the specified structure*

⊃ *action. transform the specified structure according to the specified rule*

⊃ *action. equate two specified sc-elements*

⊃ *action. include a set*

:= [make all elements of the ***Si*** set explicitly belonging to the ***Sj*** set, that is, generate the corresponding sc-arcs of belonging]

⊃ *action of generating sc-elements*

⊃ *action of generation, one of the arguments of which is some generalized structure*

⊃ *action. generate a structure isomorphic to the specified template*

⊃ *action. generate an sc-element of the specified type*

⊃ *action. generate an sc-connector of the specified type*

⊃ *action. generate an sc-node of the specified type*

⊃ *action. generate a file with the specified contents*

⊃ *action. set the specified file as the primary identifier of the specified sc-element for the specified external language*

⊃ *action. update concepts*

:= [action. replace non-basic concepts with their definition through basic concepts]

:= [action. replace some set of concepts with another set of concepts]

⊃ *action. integrate the information construction into the current state of the knowledge base*

⊃ *action. integrate the contents of the specified file into the current state of the knowledge base*

⊃ *action. translate the contents of the specified file to sc-memory*

⊃ *action. integrate the specified structure into the current state of the knowledge base*

⊃ *action. supplement the description of the past state of the ostis-system*

⊃ *action. supplement the structure describing the history of the ostis-system evolution*

⊃ *action. supplement the structure describing the history of ostis-system operation*

⊃ *action of deleting sc-elements*

⊃ *action. delete the specified sc-elements*

⊃ *action. delete sc-elements that are part of the specified structure and are not the key nodes of any sc-agents*

action. equate two specified sc-elements

:= [action. combine two specified sc-elements]

:= [action. paste two specified sc-elements together]

⇒ *subdividing**:

{• *action. physically equate two specified sc-elements*

• *action. logically equate two specified sc-elements*

}

Each ***action. equate two specified sc-elements*** can be performed as *action. physically equate two specified sc-elements* or *action. logically equate two specified sc-elements*. In the case of logical equation, the action itself is saved in the agent activity protocol with its specification, which includes a necessary indication of which elements were generated and which were deleted. In the case of physical equation, the action protocol is not saved.

Each ***action. update concepts*** denotes the transition from some group of concepts used earlier to another group of concepts that will be used instead of the first ones and will become *basic concepts*. In general, ***action. update concepts*** consists of the following steps:

- determine the concepts to be replaced based on the substitutive ones;
- make appropriate changes to the programs of sc-agents, the key nodes of which are updated concepts;
- replace all constructions in the knowledge base containing replaceable concepts, in accordance with the definitions of these concepts through the concepts that replace them;
- if necessary, *sc-elements* denoting the concepts replaced in this way can be completely deleted from

the current state of the knowledge base.

The first argument (included in the *action* sign under attribute *$1'$*) of ***action. update concepts*** is a sign for the set of *sc-nodes* denoting the replaced concepts, the second one (included in the *action* sign under attribute *$2'$*) is a sign for the set of *sc-nodes* denoting the replacing concepts. In general, either or both of these sets can be *singletons*.

action. delete the specified sc-elements

⇒ *subdividing**:

{• *action. physically delete the specified sc-elements*

• *action. logically delete the specified sc-elements*

}

Each ***action. delete the specified sc-elements*** can be performed as *action. physically delete the specified sc-elements* or *action. logically delete the specified sc-elements*. In the case of logical deletion, the action itself is saved in the agent activity protocol with its specification, which includes a necessary indication of which elements were deleted, i.e., in fact, the elements are excluded from the current state of the knowledge base. In case of physical deletion, the action protocol is not saved.

If any *sc-element* is deleted, the incident *connectives*, including *sc-connectors*, are also deleted.

To perform ***action. integrate the specified structure into the current state of the knowledge base***, it is necessary to paste *sc-elements* included in the integrated *structure* together with synonymous *sc-elements* included in the current state of the knowledge base, replace unused (for example, outdated) concepts included in the integrated *structure* on used ones (i.e. replace unused concepts with their definitions through used ones), explicitly include all elements of the integrated *structure* in the number of elements of the approved part of the knowledge base, and explicitly include all elements of the integrated *structure* in the number of elements that are part of any atomic sections of the approved fragment of the knowledge base.

B. Problems solved in sc-memory and logically atomic actions

problem solved in sc-memory

⊂ *problem*

:= [specification of the action performed in sc-memory]

:= [structure that is such a description (formulation, setting) of the corresponding action in sc-memory, which has sufficient completeness to perform the specified action]

:= [semantic neighborhood of some action in sc-memory, providing a sufficiently complete setting of this action]

action class

⊃ *action class in sc-memory*

⇐ *family of subsets**:
action in sc-memory

⇒ *subdividing**:

{• *class of logically atomic actions*

:= [class of autonomous actions]

⊃ *class of logically atomic actions in sc-memory*

• *class of logically non-atomic actions*

:= [class of non-autonomous actions]

}

Each *action* belonging to some specific *class of logically atomic actions* has two necessary properties:

- the execution of an action does not depend on whether the specified action is part of the decomposition of a more general action. When performing this action, the fact that this action precedes or follows any other actions should also not be taken into account (which is explicitly indicated using the *sequence of actions** relation);
- the specified action should be a logically integral act of transformation, for example, in semantic memory. Such an action is essentially a transaction, i.e. the result of such a transformation is a new state of the system being transformed, and the action being performed must either be performed completely or not at all, partial execution is not allowed.

At the same time, logical atomicity does not prohibit decomposing the performed action into more particular ones, each of which, in turn, will also be logically atomic.

It is proposed to divide all activities aimed at solving any problems by the ostis-system into logically atomic actions. This approach will allow for the modifiability of *ostis-systems problem solvers*, provided that the solver components correspond to *classes of logically atomic actions in sc-memory*. Such components are called sc-agents.

VII. Concept of an sc-agent and abstract sc-agent

sc-agent

:= [the only kind of *subjects* performing transformations in *sc-memory*]

:= [*subject* capable of performing *actions in sc-memory*, belonging to some specific *class of logically atomic actions*]

The logical atomicity of the actions performed by the sc-agent assumes that each sc-agent reacts to the corresponding class of situations and/or events occurring in the sc-memory and performs a certain transformation of the sc-text located in the semantic neighborhood of the processed situation and/or event. At the same time, each sc-agent generally does not contain information about which other sc-agents are currently present in the system and interacts with other sc-agents solely by forming some constructions (usually action specifications) in the shared sc-memory. As such a message, for example, a question addressed to other sc-agents in the system (it is not known in advance which one specifically) or an answer to a question posed by other sc-agents (it is not known in advance which one specifically) can serve. Thus, each sc-agent at any given time controls only a fragment of the knowledge base in the context of the problem being solved by this agent; the state of the rest of the knowledge base is generally unpredictable for the sc-agent.

Since it is assumed that copies of the same *sc-agent* or functionally equivalent *sc-agents* can work in different ostis-systems, while being physically different sc-agents, it is advisable to consider the properties and classification of non-sc-agents but classes of functionally equivalent sc-agents, which we will call ***abstract sc-agents***. Under the ***abstract sc-agent*** is understood a certain class of functionally equivalent *sc-agents*, different instances (i.e. representatives) of which can be implemented in different ways.

Each ***abstract sc-agent*** has a corresponding specification. The specification of each ***abstract sc-agent*** includes:

- specifying the key *sc-elements* of this *sc-agent*, i.e. those *sc-elements* stored in *sc-memory* that are "support points" for this *sc-agent*;
- a formal description of the conditions for initiating this *sc-agent*, i.e. those *situation* in *sc-memory* that initiate the activity of this *sc-agent*;
- a formal description of the primary initiation condition for this *sc-agent*, i.e. such a situation in *sc-memory*, which prompts the *sc-agent* to switch to the active state and start checking for its full initiation condition (for *internal abstract sc-agents*);
- a strict, complete, unambiguously understood description of the activity of this *sc-agent*, drawn up using any understandable, generally accepted means that do not require special study, for example, in natural language;
- a description of the results of executing this *sc-agent*.

Sc-agents can be classified according to various criteria. Since we can talk about a hierarchy of methods (methods of interpreting other methods) and, accordingly, a hierarchy of skills, there is a need to talk about a hierarchy of sc-agents providing interpretation of a particular method. In this context, we can talk about the hierarchy of sc-agents in two aspects:

- an *abstract sc-agent* (and, accordingly, an *sc-agent*) can uniquely correspond to a *method* (sc-agent program) describing the activity of this sc-agent. Such agents will be called *atomic abstract sc-agents*;
- sometimes, it is advisable to combine *abstract sc-agents* into collectives of such agents, which can be considered as one integral *abstract sc-agent*, from a logical point of view, working on the same principles as *atomic abstract sc-agents*, that is, reacting to events in sc-memory and describing its activities within this memory. Such an *abstract sc-agent* will not correspond to any specific *method* stored in sc-memory, but the rest of the specification of the *abstract sc-agent* (initiation condition, initial situation description, and the result of the operation of the sc-agent, etc.) remains the same, like in case of the *atomic abstract sc-agent*. Thus, we can say that the concept of atomicity/non-atomicity of an abstract sc-agent indicates how the implementation of this *abstract sc-agent* is refined – by specifying a particular method (sc-agent program) or by decomposing the *abstract sc-agent* into simpler ones. It is important to note that *non-atomic abstract sc-agents* can also be part of other, more complex *non-atomic abstract sc-agents*. Thus, a hierarchical system of abstract sc-agents is formed, in general, having a random number of levels.
- In turn, the method corresponding to the sc-agent must be interpreted by some other sc-agent of a lower level and most often by a collective of such agents, each of which is assigned its own method describing the behavior of this agent but at a lower level. Thus, we can say that the concept of atomicity/non-atomicity of abstract sc-agents is applicable within one *method description language*. In turn, we can talk about the hierarchy of *abstract sc-agents* from the point of view of the language level for description of the methods corresponding to such agents. In general, such a hierarchy can also have an unlimited number of levels, however, it is obvious that when lowering the level of the method description language, sooner or later we must approach the method description language, which will be interpreted by agents implemented at the level of the *ostis-platform*, and going even lower – to the level of the method description language, interpreted at the hardware level. Thus, in order to ensure the platform independence of *ostis-systems*, it is advisable to allocate a method description language that would be interpreted at the level of the *ostis-platform* and be the basis for the development of interpreters of higher-level languages. As such a language, an *SCP Language* (Semantic Code Programming) is proposed, which is considered as an assembler for an *associative semantic computer*.

The hierarchical approach to the description of *knowledge processing machines* and, accordingly, *problem solvers* has a number of important advantages, such as ensuring the modifiability of solvers and the convenience of their design and debugging at different levels [2], [3].

Let us consider the classification of *abstract sc-agents* according to various criteria. Classification of *abstract sc-agents* based on atomicity:

abstract sc-agent
⇒ *subdividing**:
{• *non-atomic abstract sc-agent*
• *atomic abstract sc-agent*
}

A ***non-atomic abstract sc-agent*** is understood as an *abstract sc-agent*, which is decomposed into a collective of simpler *abstract sc-agents*, each of which in turn can be both an *atomic abstract sc-agent* and ***non-atomic abstract sc-agent***. At the same time, in some variant of *decomposition of an abstract sc-agent**, the child ***non-atomic abstract sc-agent*** can become an *atomic abstract sc-agent* and be implemented accordingly.

An ***atomic abstract sc-agent*** is understood as an *abstract sc-agent*, for which the method of its implementation is specified, i.e. there is a corresponding connective of the *sc-agent program** relation.

The *SCP Language* allows setting boundaries between the logical-semantic model of the *ostis-system* and the *ostis-platform*. In this regard, we will consider *abstract sc-agents* as platform-independent ones, implemented in the SCP Language or higher-level languages based on it, and *abstract sc-agents* – as platform-dependent ones, that are implemented at the platform level (for example, in order to improve their performance). At the same time, there are a number of abstract sc-agents that cannot be implemented in principle in the *SCP Language*. This is represented in the following hierarchy:

abstract sc-agent
⇒ *subdividing**:
{• *internal abstract sc-agent*
• *effector abstract sc-agent*
• *receptor abstract sc-agent*
}
⇒ *subdividing**:
{• *abstract sc-agent that is not implemented in the SCP Language*
• *abstract sc-agent that is implemented in the SCP Language*
}
⇒ *subdividing**:
{• *abstract sc-agent for interpreting scp-programs*
• *abstract software sc-agent*
• *abstract sc-meta-agent*
}
⇒ *subdividing**:
{• *platform-dependent abstract sc-agent*
⊃ *abstract sc-agent that is not implemented in the SCP Language*
• *platform-independent abstract sc-agent*
}

abstract sc-agent that is not implemented in the SCP Language
:= [abstract sc-agent that cannot be implemented at a platform-independent level]
⇒ *subdividing**:
{• *effector abstract sc-agent*
• *receptor abstract sc-agent*
• *abstract sc-agent for interpreting scp-programs*
}

abstract sc-agent that is implemented in the SCP Language
:= [abstract sc-agent that can be implemented at a platform-independent level]
⇒ *subdividing**:
{• *abstract sc-meta-agent*
• *abstract software sc-agent implemented in the SCP Language*
}

abstract software sc-agent
⇒ *subdividing**:
{• *effector abstract sc-agent*
• *receptor abstract sc-agent*
• *abstract software sc-agent implemented in the SCP Language*
}

atomic abstract sc-agent
⇒ *subdividing**:
{• *platform-independent abstract sc-agent*
• *platform-dependent abstract sc-agent*
}

Platform-independent abstract sc-agents include *atomic abstract sc-agents* implemented in the basic programming language of the OSTIS Technology, i.e. in the *SCP Language*.

When describing ***platform-independent abstract sc-agents***, platform independence is understood as platform independence from the point of view of the OSTIS Technology, i.e. implementation in a specialized programming language focused on processing semantic networks (*SCP Language*), since *atomic sc-agents* implemented in the specified language can be freely transferred from one ostis-platform to another. At the same time, programming languages that are traditionally considered platform-independent in this case cannot be considered as such.

There are *sc-agents* that fundamentally cannot be implemented at a platform-independent level, for example, the actual *sc-agents* for interpreting *sc-models* or receptor and effector *sc-agents* that provide interaction with the external environment.

Platform-dependent abstract sc-agents include *atomic abstract sc-agents* implemented below the level of sc-models, i.e. not in the *SCP Language* but in some other program description language.

Each ***internal abstract sc-agent*** denotes a class of *sc-agents* that react to events in *sc-memory* and perform transformations exclusively within the same *sc-memory*.

Each ***effector abstract sc-agent*** denotes a class of *sc-agents* that react to events in *sc-memory* and perform transformations in an environment external to this *ostis-system*.

Each ***receptor abstract sc-agent*** designates a class of *sc-agents* that react to events in the environment external to this *ostis-system* and perform transformations in the memory of this system.

Each ***abstract sc-agent that is not implemented in the SCP Language*** must be implemented at the level of the *ostis-platform*, including hardware one. Such *abstract sc-agents* include abstract sc-agents for interpreting scp-programs, as well as effector and receptor abstract sc-agents.

Each ***abstract sc-agent implemented in the SCP Language*** can be implemented in the *SCP Language*, that is, at the platform-independent level, but, if necessary, it can also be implemented at the platform level, for example, in order to improve performance.

Abstract sc-agents for interpreting scp-programs include *abstract sc-agents* that are not implemented at the platform-independent level, providing interpretation of *scp-programs* and *scp-meta-programs*, including the creation of *scp-processes*, the actual interpretation of *scp-operators*, as well as other auxiliary actions. In fact, agents of this class ensure the operation of sc-agents of higher levels (software sc-agents and sc-meta-agents) implemented in the SCP Language, in particular, ensure that these agents comply with the general principles of synchronization.

Abstract software sc-agents includes all *abstract sc-agents* that provide the basic functionality of the system, that is, its ability to solve certain problems. Agents of this class should work in accordance with the general principles of synchronizing the activities of subjects in sc-memory.

The purpose of ***abstract sc-meta-agents*** is to coordinate the activities of *abstract software sc-agents*, in particular, solving the problem of interlocks. Agents of this class can be implemented in the SCP Language, however, other principles are used to synchronize their activities, respectively, to implement such agents, a different level of the SCP Language is required, the typology of which operators is completely similar to the typology of scp-operators, however, these operators have different operational semantics, taking into account differences in the principles of synchronization (working with *locks**). Programs of such a language will be called *scp-meta-programs*, corresponding to them *processes in sc-memory – scp-meta-processes*, operators – *scp-meta-operators*.

decomposition of an abstract sc-agent*
∈ *decomposition relation*

The ***decomposition of an abstract sc-agent**** relation interprets *non-atomic abstract sc-agents* as collectives of simpler *abstract sc-agents* interacting through *sc-memory*.

In other words, ***decomposition of an abstract sc-agent**** into *abstract sc-agents* of a lower level clarifies one of the possible approaches to the implementation of this *abstract sc-agent* by building a collective of simpler *abstract sc-agents*.

sc-agent
:= [agent on sc-memory]
⊂ *subject*
⇒ *family of subsets**:
abstract sc-agent

An ***sc-agent*** is understood as a concrete instance (from a set-theoretic point of view, an element) of some *atomic abstract sc-agent* operating in any particular intelligent system.

Thus, each *sc-agent* is a subject capable of performing some class of similar actions either only on *sc-memory* or on sc-memory and the external environment (for effector*sc-agents*). Each such action is initiated either by a state or situation in sc-memory, or by a state or situation in the external environment (for receptor sc-agents-sensors) corresponding to the initiation condition of the *atomic abstract sc-agent*, which instance is the specified *sc-agent*. In this case, an analogy can be drawn between the principles of object-oriented programming, considering an *atomic abstract sc-agent* as a class, and a specific *sc-agent* as an instance, a specific implementation of this class.

Interaction of *sc-agents* is carried out only through *sc-memory*. As a consequence, the result of the operation of any *sc-agent* is some change in the state of *sc-memory*, i.e. the deletion or generation of any *sc-elements*.

In general, one *sc-agent* can explicitly transfer control to another *sc-agent* if this *sc-agent* is known a priori. To do this, each *sc-agent* in *sc-memory* has an *sc-node* denoting it, with which it is possible to associate a specific situation in the current state of the knowledge base that the initiated *sc-agent* must process.

However, it is not always easy to determine the *sc-agent* which should take control from a given *sc-agent*, and therefore the situation described above occurs extremely rarely. Moreover, sometimes the condition for initiating the *sc-agent* is the result of the activity of an unpredictable group of *sc-agents*, as well as the same construction can be the condition for initiating an entire group of *sc-agents*.

At the same time, not *sc-agent programs** communicate through *sc-memory* but the *sc-agents* themselves described by these programs.

In the process of work, the *sc-agent* can generate auxiliary *sc-elements* for itself, which it deletes after completing the act of its activity (these are auxiliary *structures* that are used as "information scaffolds" only during the execution of the corresponding act of activity and are deleted after the performance of the act).

sc-agent
⊃ *active sc-agent*
⇒ *first domain**:
- *key sc-elements of the sc-agent**
- *sc-agent program**
- *primary initiation condition**
- *initiation condition and result**

An ***active sc-agent*** is understood as an *sc-agent* of the ostis-system, which reacts to events corresponding to its initiation condition and, as a consequence, its *primary initiation condition**. The *sc-agents* that are not included in the set of ***active sc-agents*** do not respond to any events in *sc-memory*.

The connectives of the ***key sc-elements of the sc-agent**** relation link together the *sc-node*, denoting an *abstract sc-agent*, and the *sc-node*, denoting the set of *sc-elements*, which are key for a given *abstract sc-agent*, that is, given *sc-elements* are explicitly mentioned within programs implementing this *abstract sc-agent*.

The connectives of the ***sc-agent program**** relation link together the *sc-node*, denoting an *atomic abstract sc-agent*, and the *sc-node*, denoting a set of programs implementing the specified *atomic abstract sc-agent*. In the case of *platform-independent abstract sc-agent*, each connective of the *sc-agent program** relation connects the *sc-node* denoting the specified *abstract sc-agent* with a set of *scp-programs* describing the activities of this *abstract sc-agent*. This set contains one *agent scp-program* and a random number (maybe none) of *scp-programs* that are necessary to execute the specified *agent scp-program*.

In the case of the *platform-dependent abstract sc-agent*, each connective of the *sc-agent* program* relation links the *sc-node* denoting the specified *abstract sc-agent* with a set of files containing the source texts of the program in some external programming language that implements the activity of this *abstract sc-agent*.

The connectives of the ***primary initiation condition**** relation link together the *sc-node*, denoting an *abstract sc-agent*, and a binary oriented pair describing the primary initiation condition of this *abstract sc-agent*, i.e. such a specification of the *situations* in *sc-memory*, the occurrence of which prompts the *sc-agent* to switch to the active state and start checking for its full initiation condition.

The first component of this oriented pair is the sign of some class of *elementary events in sc-memory**, for example, the *event of adding an sc-arc going out of a given sc-element**.

In the general case, the second component of this oriented pair is a random *sc-element*, with which the specified type of event in *sc-memory* is directly associated, i.e., for example, the *sc-element*, from which the generated or deleted *sc-arc* or *file*, the contents of which have been changed, goes out, or in which this sc-arc or the file come.

After an event occurs in *sc-memory*, all *active sc-agents* are activated, the ***primary initiation condition**** of which corresponds to the event that occurred.

The connectives of the ***initiation condition and result**** relation link together the *sc-node*, denoting an *abstract sc-agent*, and a binary oriented pair linking the initiation condition for this *abstract sc-agent* and the results of executing this instance of the given *sc-agent* in any particular system.

The specified oriented pair can be considered as a logical implication connective, while the universality quantifier is implicitly imposed on *sc-variables* present in both parts of the connective and the existence quantifier is implicitly imposed on *sc-variables* present either only in the premise or only in the conclusion.

The first component of the specified oriented pair is a logical formula describing the initiation condition for the described *abstract sc-agent*, that is, a construction whose presence in *sc-memory* prompts the *sc-agent* to begin work on changing the state of *sc-memory*. This logical formula can be both atomic and non-atomic, in which the usage of any logical language connectives is allowed.

The second component of the specified oriented pair is a logical formula describing the possible results of the execution of the described abstract *sc-agent*, that is, a description of the changes in the state of *sc-memory* made by it. This logical formula can be both atomic and non-atomic, in which the usage of any logical language connective is allowed.

description of the behavior of an sc-agent
⊂ *semantic neighborhood*

The ***description of the behavior of an sc-agent*** is a *semantic neighborhood* describing the activity of an *sc-agent* to some degree of detail, however, such a description must be strict, complete, and unambiguously understood. Like any other *semantic neighborhood*, the ***description of the behavior of an sc-agent*** can be translated into any understandable, generally accepted means that do not require special study, for example, into natural language.

The described *abstract sc-agent* is included in the corresponding ***description of the behavior of an sc-agent*** under the *key sc-element′* attribute.

VIII. Principles of synchronizing the activities of sc-agents

A. Clarification of the typology of processes in sc-memory, concepts of locks and locks classification

The concepts of an *action in sc-memory* and a *process in sc-memory* (information process performed by an agent in semantic memory) are synonymous, since all processes occurring in sc-memory are conscious and are performed by some sc-agents. Nevertheless, when it comes to synchronizing the execution of any transformations in the memory of a computer system, it is accepted in the literature to use the terms "process" and "interaction of processes" [15], [16], in connection with which we will use this term when describing the principles of synchronizing the activities of sc-agents when they perform parallel processes in sc-memory.

process in sc-memory
⇒ *subdividing**:
  {• *process in sc-memory corresponding to a platform-dependent sc-agent*
   • *scp-process*
    ⇒ *subdividing**:
     {• *scp-process that is not an scp-meta-process*
      • *scp-meta-process*
     }
  }

process in sc-memory corresponding to a platform-dependent sc-agent
⇒ *subdividing**:
  {• *process in sc-memory that corresponds to a platform-dependent sc-agent and is not an action of an abstract scp-machine*
   • *action of an abstract scp-machine*
    ⊃ *action of scp-program interpretation*
  }

To synchronize the execution of *processes in sc-memory*, it is proposed to use a locking mechanism based on existing algorithms for synchronizing information processes in traditional systems [15], [16]. As a possible direction for the development of this approach, it is possible to indicate the ideas of lock-free algorithms that are gaining popularity [17].

The ***lock**** relation connects the signs of *actions in sc-memory* with the signs of *structures* (situational ones) that contain elements that are blocked for the duration of performing this action or for some part of this period. Each such *structure* belongs to one of the *lock types*.

The first component of the connective of the ***lock**** relation is the sign of an *action in sc-memory*, the second is the sign of the blocked *structure*.

lock*
∈ *binary relation*

lock type
∋ *full lock*
∋ *lock on any change*
∋ *lock on deletion*

The ***lock type*** set contains all possible lock classes, i.e. *structures* containing *sc-elements* blocked by some *sc-agent* for the duration of performing some *action in sc-memory*.

Each *structure* belonging to the ***full lock*** set contains *sc-elements*, viewing and modification (deletion, addition of incident *sc-connectors*, deletion of the *sc-elements* themselves, changing the contents in the case of a file) which are prohibited to all *sc-agents*, except for the *sc-agent* itself, which performs the corresponding *action in sc-memory* associated with it by the *lock** relation.

In order to exclude the possibility of implementing *sc-agents*, which can make changes to the constructions describing the locks of other *sc-agents*, all elements of these constructions, including the sign of the *structure* containing the blocked *sc-elements* (belonging to both the ***full lock*** set and any other *lock type*) and the connectives of the *lock** relation linking this *structure* and a specific *action in sc-memory* are added to the ***full lock***, corresponding to the given *action in sc-memory*. Thus, each ***full lock*** corresponds to an affiliation loop linking its sign to itself.

Each *structure* belonging to the ***lock on any change*** set contains *sc-elements*, modification (physical deletion, addition of incident *sc-connectors*, physical deletion of *sc-elements*, changing the contents in the case of a file), which is prohibited to all *sc-agents*, except for the *sc-agent* itself, which performs the corresponding *action in sc-memory* associated with it by the *lock** relation. However, viewing (reading) of these *sc-elements* by any *sc-agent* is not prohibited.

Each *structure* belonging to the ***lock on deletion*** set contains *sc-elements*, the deletion of which is prohibited to all *sc-agents*, except for the *sc-agent*, which performs an action corresponding to this structure *in sc-memory*, associated with it by the *lock** relation. However, it is not prohibited to view (read) these *sc-elements* by any *sc-agent*, adding incident sc-connectors.

B. Principles of working with locks

Let us consider the principles of working with locks:

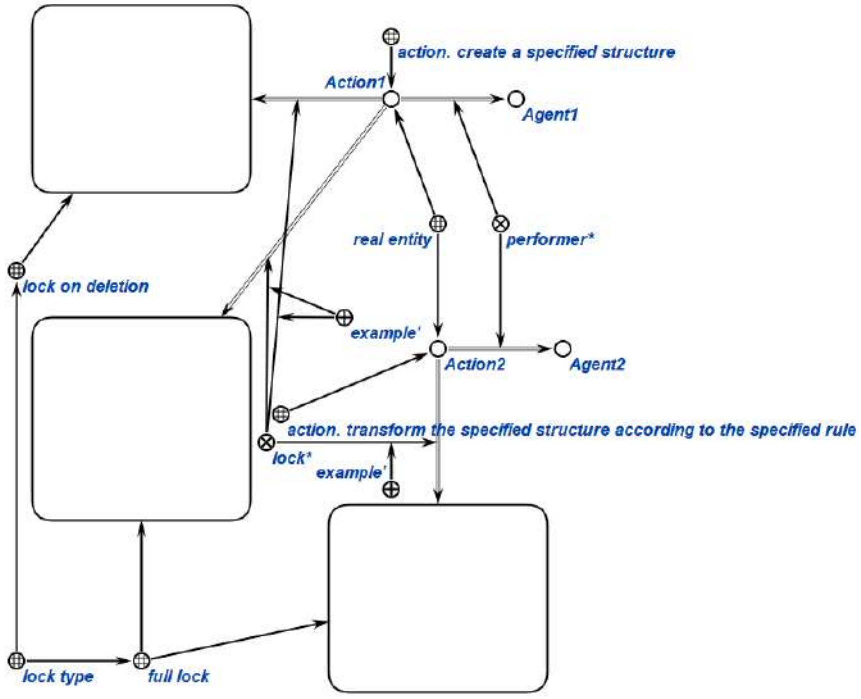

action. create a specified structure
Action1
Agent1
real entity
performer*
lock on deletion
example'
Action2
Agent2
action. transform the specified structure according to the specified rule
lock*
example'
lock type
full lock

Figure 1. An example of using locks

- at any given moment, only one lock of each type can correspond to one process in sc-memory;
- at any given time, only one lock can correspond to one process in sc-memory, set on some specific sc-element;
- at the end of any process execution in sc-memory, all the locks set by it are automatically deleted;
- to increase the efficiency of the system as a whole, each process must block the minimum required set of sc-elements at any given time, removing the lock from each sc-element as soon as it becomes possible (safe);
- In the case when more particular subprocesses are explicitly allocated within the *process in sc-memory* (using the *temporal part*, sub-action*, action decomposition*, etc. relations*), then each such subprocess from the point of view of synchronizing execution is considered as an independent process, which can correspond to all necessary locks.
  - all child processes in sc-memory have access to the locks of the maternal process in the same way as if they were locks corresponding to each of such child processes;
  - in turn, the maternal process does not have any privileged access to sc-elements blocked by child processes and works with them in the same way as any other process in sc-memory. The exception is sc-elements denoting the child processes themselves, since the maternal process must be able to control the child one, for example, suspending or terminating their execution;
  - all child processes in relation to each other work the same way as in relation to any other processes;
  - in the case when the maternal process suspends execution (becomes a *deferred action*), all of its child processes also suspend execution. In turn, suspending one of the child processes in general does not explicitly initiate the stopping of the entire maternal process and, accordingly, other child processes.

Let us consider the principles of working with *full locks*:

- if the sc-element incident to some sc-connector gets into any full lock, then this sc-connector itself is also considered blocked by the same lock by default. The contrary is generally not true, since part of the sc-connectors incident to some sc-element may be completely blocked, while this element itself will not be blocked. This situation is typical, for example, for sc-nodes denoting classes of concepts;
- each process in sc-memory can freely modify or delete any sc-elements that get into the full lock corresponding to this process.

The principles of working with *full locks*, on the one hand, are the simplest, since all processes, except for the one who set such a lock, do not have access to the blocked sc-elements, and conflicts cannot arise. On the other hand, the frequent usage of locks of this type can lead to the case when the system will not be able to fully use its knowledge and give incomplete or even incorrect answers to the questions posed.

Let us consider the principles of working with *locks on any change* and *locks on deletion*:

- only one lock of the same type can be set on the same sc-element at one time, but different processes can simultaneously set two different locks types on the same element. This concerns the case when the first process has set a lock on deletion on some sc-element and the second process then sets a lock on any change. In other cases, a lock conflict occurs;
- setting a lock of any type is also considered a change, so if a lock on any change was set on some sc-element, then another process will not be able to set a lock of any type on the same sc-element until the first process deletes its own;
- if a lock on deletion is set on some sc-connector, then by default the same lock is set on sc-elements that are incident to this sc-connector, since deleting these elements will lead to the deletion of this connector.

process in sc-memory
:= [action in sc-memory]
⇒ *subdividing**:
Classification of processes in sc-memory in terms of synchronizing their execution
= {• *action of searching for sc-elements*
• *action of generating sc-elements*
• *action of deleting sc-elements*
• *action of setting a lock of some type on some sc-element*
• *action of removing the lock from some sc-element*
}

In some cases, in order to ensure synchronization, it is necessary to combine several elementary actions on sc-memory into one indivisible action (***transaction in sc-memory***), for which it is guaranteed that no third-party process will be able to read or modify the sc-elements involved in this action, until the action completes. At the same time, unlike a situation with a full lock, a process, trying to access such elements, does not continue execution as if these elements simply did not exist in sc-memory but waits for the transaction to complete, after which it can perform any actions with these elements according to the general principles of process synchronization. The problem of ensuring transactions cannot be solved at the SC-code level and requires the implementation of such indivisible actions at the level of the *ostis-platform*.

If an *action of searching for sc-elements* is performed, all sc-elements found and saved within any process get into the corresponding *lock on any change* for this process. Thus, the integrity of the fragment of the knowledge base with which some process is working in sc-memory is guaranteed. In this case, the search and automatic setting of such a lock should be implemented as a *transaction in sc-memory*.

This approach also allows avoiding a situation where one process has blocked some sc-element on any change, and the second process is trying to generate or delete an *sc-connector* incident to this *sc-element*. In this case, the second process will have to first find and lock the specified *sc-element* on any change, which will cause a lock conflict (*interlock**).

In the case of generation of any sc-element within a certain process, it automatically gets into a full lock corresponding to this process. At the same time, the generation and automatic setting of such a lock should be implemented as a *transaction in sc-memory*. If necessary, the generated elements can be deleted (i.e. their temporary existence will not affect the activities of other processes at all) or unblocked when information is generated that may have some value in the future.

If any process tries to set a lock of any type on any sc-element already blocked by some other process, then, on the one hand, the lock cannot be set until another process unlocks the specified sc-element; on the other hand, in order to provide the possibility of searching and eliminating *interlocks*, it is necessary to explicitly indicate the fact that some process wants to access some sc-element blocked by another process. In order to be able to specify which processes are trying to block an already blocked *sc-element*, it is proposed, along with the *lock** relation, to use the *planned lock** relation, completely analogous to the *lock** relation.

The described mechanism also regulates the search processes, since the searching and saving of some sc-element involves the setting of a *lock on any change*. In addition, it should be taken into account that a *lock on any change* can be set on one sc-element after the *lock on deletion* corresponding to another process. In this case, there is no need to use the *planned locks** relation.

The action of checking for the presence of a lock on some sc-element and, depending on the result of the check, the setting of the lock or the planned lock (indicating the priority, if necessary) should be implemented as a transaction.

planned lock*
⊂ *lock**

The process to which the *planned lock** is assigned suspends execution until the already set locks are removed, after which the *planned lock** becomes a real *lock**, and the process continues execution in accordance with the general rules.

lock priority*

⇒ *scope of definition**:
*planned lock**

In the case when several processes are planning to set a lock on the same sc-element at once, the *lock priority** relation is used, linking the *planned lock** relation pairs. As a rule, the lock priority is determined by which of the processes previously tried to set a lock on the given sc-element, although in general the priority can be set or changed depending on additional criteria.

In the case of an attempt to delete some sc-element by some process, deletion can be carried out only if no lock is set (and is not planned to be set) on this sc-element by any other process.

In other cases, it is necessary to ensure that all processes working with this sc-element are completed correctly, and only then delete it physically.

To implement this possibility, each process can be matched with a set of sc-elements that are deleted by this process.

The action of checking for locks or planned locks on the deleted sc-element and actually deleting it or adding it to the set of deleted sc-elements for the corresponding process should be implemented as a transaction.

deleted sc-elements*

⇒ *first domain**:
process in sc-memory

Sc-elements that have got into the set of deleted sc-elements of some process in sc-memory are available to processes that have already set (or plan to set) locks on these sc-elements earlier (before attempting to delete it), and for all other processes these sc-elements are already considered deleted. A process trying to delete an sc-element suspends its execution until all processes, which have blocked and plan to block this sc-element, unlock it. In general, one sc-element can be included in the sets of deleted elements simultaneously for several processes, in this case, all such processes will simultaneously continue execution after removing all locks from this sc-element. If the deletion is attempted by one of the processes that has already set a lock on the specified sc-element, then the algorithm of actions remains the same – the sc-element is added to the set of sc-elements being deleted by this process and will be physically deleted as soon as all other processes that have set a lock on this sc-element remove them.

Let us consider the algorithm for removing the lock from some sc-element:

1) if one or more *planned locks** are set on this sc-element, then the first of them by priority (or the only one) becomes a *lock**, the corresponding process continues execution (becomes a real entity); the connective of the execution priority relation corresponding to the remote connective of the *planned lock** relation is also deleted, i.e. the priority is shifted by one position;
2) if there are no *planned locks** set on this sc-element, but it gets into the set of deleted sc-elements for one or more processes, then the given sc-element is physically deleted and the processes, suspended before its deletion, continue their execution (become real entities);
3) if the planned locks are not set on this sc-element and it is not included in the set of deleted ones for any process, then the lock is simply removed without any additional changes.

transaction in sc-memory

⇒ *subdividing**:
{
- *searching for some construction in sc-memory and automatic setting a lock on any change to the found sc-elements*
- *generating some sc-element and automatic setting of a full lock on it*
- *checking for the presence of a lock on some sc-element and, depending on the result of the check, setting a lock or a planned lock*
- *checking for the presence of locks or planned locks on the deleted sc-element and actually deleting it or adding it to the set of deleted sc-elements for the corresponding process*
- *removing the lock from a given sc-element and, if necessary, setting the first in priority planned lock or deleting this sc-element if it is included in the set of deleted sc-elements for some process*
- *searching for subprocesses of a process and adding them to a set of deferred actions in the case of adding the process itself to this set*
- *searching for subprocesses of a process and deleting them from the set of deferred actions if the process itself is deleted from this set*

}

C. Principles of synchronizing sc-agents implemented at the platform-independent level

When implementing *abstract software sc-agents* in the *SCP Language*, compliance with all the principles of synchronization of processes corresponding to these sc-agents is ensured at the level of *sc-agents for interpreting scp-programs*, i.e. by means of the *ostis-platform*. When implementing *abstract software sc-agents* at the platform level, compliance with all synchronization principles is assigned, firstly, directly to the agent developer and, secondly, to the platform developer. For example, the platform can provide access to elements stored in sc-memory through some programming interface that already takes into account the principles of working with locks, which will save the agent developer from having to take into account all these principles manually.

In addition, a number of specific principles of operation of *abstract software sc-agents*, implemented in the *SCP Language*, are highlighted:

- as a result of the appearance in sc-memory of some construction that satisfies the condition of initiating some *abstract sc-agent* implemented using the *SCP Language*, an *scp-process* is generated and initiated in *sc-memory*. As a template for generation, an *agent scp-program* is used, corresponding to this *abstract sc-agent*.
- each such *scp-process* corresponding to some *agent scp-program* can be associated with a set of structures describing locks of various types. Thus, synchronization of interaction of parallel *scp-processes* is carried out in the same way as in the case of any other *actions in sc-memory*.
- despite the fact that each *scp-operator* is an atomic action in sc-memory, which is a sub-action within the entire *scp-process*, locks corresponding to one operator are not introduced to avoid the lengthiness and excess of additional system constructions created when executing some *scp-process*. Instead of it, locks that are common to the entire *scp-process* are used. Thus, *agents for interpreting scp-programs* work only taking into account the locks common to the entire interpreted *scp-process*.
- processes describing the activity of agents for interpreting *scp-programs* are usually not created, therefore, their corresponding locks are not introduced. Since such agents work with a unique scp-process and their number is limited and known, then the usage of locks for their synchronization is not required.
- if the *scp-process* is suspended (is added to the set of *deferred actions*), in accordance with the general synchronization rules, all its child processes must also be suspended. In this regard, all *scp-operators*, which at this moment are *real entities*, become *deferred actions*.
- in order to avoid undesirable changes in the body of the *scp-process*, the entire construction generated on the basis of some *scp-program* (the entire *sc-text* describing the decomposition of the *scp-process* into *scp-operators*) must be added to the *full lock* corresponding to this *scp-process*.
- if necessary, the corresponding *scp-operators* of the *scp-operator for lock control* class are used to unlock or lock some construction by some lock type.
- after completing the execution of some scp-process, its text is usually deleted from *sc-memory* and all blocked constructions are released (signs of structures that denoted locks are destroyed).
- as a rule, the particular *action class* corresponding to a specific *scp-program* is not explicitly introduced, but the more general *scp-process* class is used, except in cases when the introduction of a special *action class* is necessary for some other reasons.

In general, the entire locking mechanism can be described both at the SC-code level (to increase the level of platform independence) and, if necessary, can be implemented at the *ostis-platform* level, for example, to improve performance. To do this, a unique table, containing a list of blocked elements with an indication of the lock type at each time, can be assigned to each process executed in sc-memory at the lower level.

D. Example of the operation of the locking mechanism

Let us consider an example of using the described mechanism.

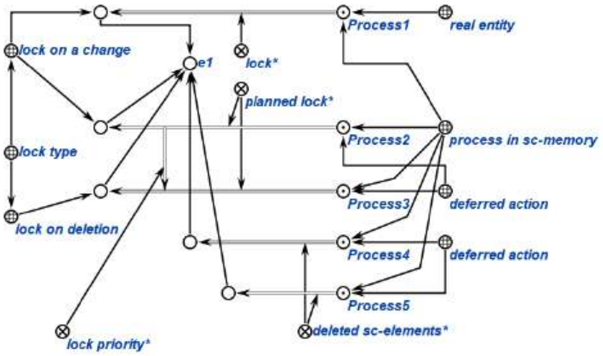


Figure 2. An example of using planned locks

In this example, *Process1* works directly with the sc-element ***e1***,*Process2* and *Process3* plan to set a lock on any change and a lock on deletion, respectively, at the same time, *Process2* tried to set its lock before *Process3*, therefore, according to the direction of the connective of the *lock priority** relation, its lock will be set earlier. *Process4* and *Process5* are waiting for all locks and planned locks to be removed, after which ***e1*** will be deleted, and *Process1* and *Process2* will continue their execution. No other planned locks can be set anymore, since ***e1*** got into a set of deleted sc-elements of at least one process and, in accordance with the rules set out above, all other processes except *Process1–Process5* can no longer access this sc-element. The executed process belongs to the real entity set, suspended – to the deferred action set.

After *Process1* has unlocked sc-element ***e1***, this element will be locked by *Process2*, and *Process2* will continue execution. *Planned lock** set by *Process2*, becomes a regular *lock**.

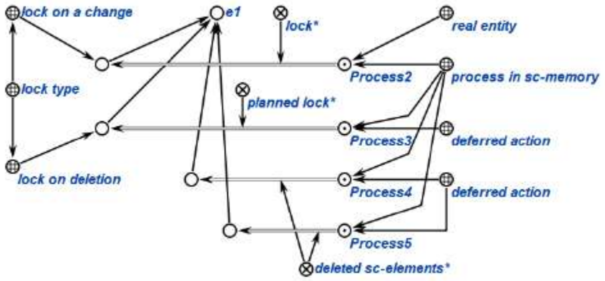


Figure 3. An example of using planned locks (continued)

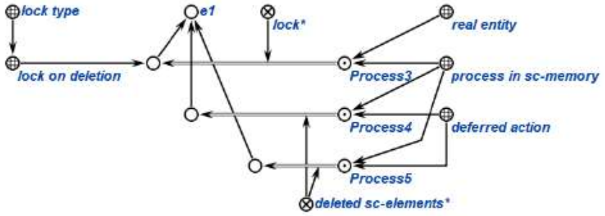


Figure 4. An example of using a lock on deletion

After *Process2* has unlocked sc-element ***e1***, this element will be locked by *Process3*, and *Process3* will continue execution.

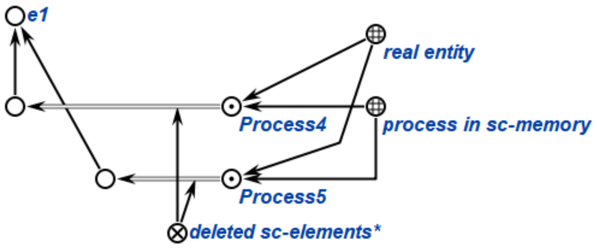


Figure 5. The sc-elements to be deleted

When all processes remove the locks from sc-element ***e1***, it can be physically deleted, and *Process4* and *Process5* will continue execution.

Depending on the specific *lock types* set by parallel processes on some sc-elements and what specific actions with these *sc-elements* are supposed to be performed further within these processes, the interlock situations are possible when each of these processes will wait for the second process to remove the lock from the desired *sc-element*, without removing the lock set by itself from the *sc-element*, access to which is required by the second process.

In the case when at least one of the locks is a *full lock*, an interlock situation cannot occur, since *sc-elements* that have got into the *full lock* of some *scp-process* are not available to other *scp-processes*, even for reading, and, thus, the rest of the *scp-processes* will work as if the blocked *sc-elements* are simply missing in the current state of *sc-memory*.

In cases where none of the set locks is a *full lock*, interlocks may occur.

Elimination of the *interlock* is impossible without the intervention of a specialized *sc-meta-agent*, which has the right to ignore the locks set by other processes.

In general, the problem of a specific interlock can be solved by performing the following steps by a specialized *sc-meta-agent*:

- rollback of several operations performed by one of the processes involved in the interlock by as many steps back as necessary so that the second process gets access to the necessary *sc-elements* and can continue execution;
- waiting for the execution of the second process until it completes or removes all locks from *sc-elements* that the first process needs to access;
- repeated execution of canceled operations within the first process and continuation of its execution but taking into account the changes in memory made by the second process.

For *sc-meta-agents*, all sc-elements, including those describing locks, planned locks, etc., are completely equivalent to each other in terms of access to them, i.e. any *sc-meta-agent* has access to any sc-elements, even those that have got into a full lock for any other process. This is necessary so that *sc-meta-agents* can identify and fix various problems, for example, the interlock problem described above.

Thus, the problem of synchronizing the activities of *sc-meta-agents* requires the introduction of additional rules.

We will divide this problem into two more specific ones:

- ensuring synchronization of the activities of *sc-meta-agents* among themselves;
- ensuring synchronization of the activities of *sc-meta-agents* and *software sc-agents*.

The first problem is proposed to be solved by prohibiting parallel execution of *sc-meta-agents*. Thus, at any given time within one *ostis-system*, there can be only one process corresponding to the *sc-meta-agent* and being the *real entity*.

The second problem is proposed to be solved by introducing additional privileges for *sc-meta agents* when accessing any sc-element. One rule is enough for this:

If a certain sc-element has become used within a process corresponding to the *sc-meta-agent* (for example, it has become an element of at least one scp-operator included in this process), then all processes, into the locks corresponding to which the specified sc-element gets, become deferred actions (suspend execution). As soon as the specified sc-element ceases to be used within the process corresponding to the *sc-meta-agent*, all processes suspended for this reason continue execution.

The considered limitations do not significantly deteriorate the performance of the ostis-system, since *sc-meta-agents* are designed to solve a fairly narrow class of problems, which, as the experience of practical developing prototypes of various *ostis-systems* has shown, arise quite rarely.

It is worth noting that there may be a situation in which the execution of some process in sc-memory is interrupted due to an error. In this case, there is a possibility that the lock set by this process will not be removed until the sc-meta-agent that has detected a similar situation does that. However, this problem can only be partially solved at the sc-model level, for cases when an error occurs during the interpretation of the scp-program, is tracked by the scp-interpreter, and a corresponding construction is formed in memory that reports the problem to the sc-meta-agent. Cases where an error has occurred at the scp-interpreter or sc-storage levels should be considered at the ostis-platform level.

IX. Basic programming language of ostis-systems

The allocation of the Basic programming language for ostis-systems allows for a clear separation of the level of methods and, accordingly, the skills of the ostis-system, which can be fully described at the level of the knowledge base, and lower-level skills that provide interpretation of these higher-level skills. In other words, the allocation of such a language allows for the platform independence of ostis-systems, both in the case of a software implementation of the ostis-platform and in the case of an *associative semantic computer*.

As a basic language for describing programs for processing texts of the *SC-code*, the *SCP Language* is proposed.

The *SCP Language* is a graph procedural programming language designed for efficient processing of *sc-texts*. The *SCP Language* is a parallel asynchronous programming language.

SCP Language
:= *frequently used sc-identifier**:
[scp-program]

The data representation language for texts of the *SCP Language* (*scp-programs*) is the *SC-code* and, accordingly, any variants of its external representation. The *SCP Language* is built on the basis of the *SC-code*, as a result of which *scp-programs* can be part of the processed data for *scp-programs*, including in relation to themselves. Thus, the *SCP Language* provides the ability to build reconfigurable programs. However, in order to be able to reconfigure the program directly in the process of its interpretation, it is necessary at the level of the interpreter of the *SCP Language (Abstract scp-machine)* ensure the uniqueness of each executable copy of the source program. Such an executable copy generated on the basis of the *scp-program* will be called an *scp-process*. The inclusion of the sign of some *action in sc-memory* in the set of *scp-processes* guarantees the fact that only the signs of elementary actions (*scp-operators*) will be present in the decomposition of this action, which can be interpreted by the implementation of the *Abstract scp-machine* (interpreter of scp-programs).

The *SCP Language* is considered as an assembler for an *associative semantic computer* [3].

Abstract scp-machine
∈ *scp-machine*
⇐ *generalized model**:
scp-interpreter

The *basic model for processing sc-texts* includes the *Subject domain of the Basic programming language of ostis-systems*, that is, a description of the syntax and denotational semantics of the SCP Language, as well as a description of the *Abstract scp-machine* that is a model of the *scp-interpreter*, which should be part of the *ostis-platform* (although in general there can exist platform variants that do not contain such an interpreter, which, however, will not allow using the advantages of the proposed basic model).

Let us consider the key features and advantages of the *Basic model for processing sc-texts*:

- The texts of the *SCP Language* programs are written using the same unified semantic networks as the processed information, so we can say that the *Syntax of the SCP Language* at the basic level is the same as the *Syntax of the SC-code*.
- An approach to interpreting *scp-programs* involves creating a unique *scp-process* at each call of the *scp-program*.
- Several independent *sc-agents* can be executed simultaneously in shared memory, while different copies of *sc-agents* can be executed on different servers, due to the distributed implementation of the ostis-platform. Moreover, the *SCP Language* allows making parallel asynchronous calls to subprograms with subsequent synchronization and even executing operators in parallel within a single *scp-program*.
- The transfer of the *sc-agent* from one system to another consists in a simple transfer of a fragment of the knowledge base, without any additional operations depending on the interpretation platform.
- The fact that the specifications of *sc-agents* and their programs can be written in the same language as the processed knowledge significantly reduces the list of specialized tools intended for designing knowledge processing machines and simplifies their development by using more universal components.

- The fact that a unique *scp-process* is created for the interpretation of the *scp-program* makes it possible to optimize the execution plan before its implementation and even directly during execution without the potential danger of ruining the general universal algorithm of the entire program. Moreover, such an approach to the design and interpretation of programs allows talking about the possibility of creating self-reconfigurable programs.

A. Concept of an scp-program

scp-program
⊂ *program in sc-memory*
⊃ *agent scp-program*

Each ***scp-program*** is a *generalized structure* describing one of the decomposition options for actions of some class performed in sc-memory. The sign of the *sc-variable* corresponding to a specific decomposable action is a *key sc-element'* within the ***scp-program***. It is also explicitly indicated that this sign belongs to the set of *scp-processes*.

Thus, each ***scp-program*** describes in a generalized form the decomposition of some *scp-process* into interrelated *scp-operators*, indicating, if any, arguments for this *scp-process*.

Agent scp-programs are a special case of *scp-programs* in general, however, they deserve separate consideration, since they are used most often. *Scp-programs* of this class are implementations of programs of knowledge processing agents and have a rigidly fixed set of parameters. Each such program has exactly two *in-parameters'*. The value of the first parameter is the sign of a binary oriented pair, which is the second component of the connective of the *primary initiation condition** relation for an abstract *sc-agent*, the set of *sc-agent programs** of which includes the considered ***agent scp-program***, and in fact, it describes a class of events, to which the specified sc-agent responds.

The value of the second parameter is an *sc-element*, which is directly associated with the event, as a result of which the corresponding *sc-agent* was initiated, i.e., for example, generated or deleted *sc-arc* or *sc-edge*.

Let us consider the principles of implementing *abstract sc-agents implemented in the SCP Language*:

- general principles of the organization of interaction between *sc-agents* and users of the *ostis-system* through a shared *sc-memory*;
- as a result of the appearance in sc-memory of some construction that satisfies the condition of initiating some *abstract sc-agent* implemented using the *SCP Language*, the *scp-process* is generated and initiated in *sc-memory*. As a template for generation, an *agent scp-program* is used, specified in the set of programs of the corresponding *abstract sc-agent*;
- each such *scp-process* corresponding to some *agent scp-program* can be associated with a set of structures describing locks of various types. Thus, synchronization of interaction of parallel *scp-processes* is carried out in the same way as in the case of any other *actions in sc-memory*;
- Within the *scp-process*, child *scp-processes* can be created, but synchronization between them, if necessary, is carried out by introducing additional internal locks. Thus, each *scp-process* from the point of view of *processes in sc-memory* is atomic and complete act of activity of some *sc-agent*;
- in order to avoid undesirable changes in the body of the *scp-process* itself, the entire structure generated on the basis of some *scp-program* (the entire text of the *scp-process*) should be added to the *full lock* corresponding to this *scp-process*;
- all constructions generated during the execution of the *scp-process* automatically get into the *full lock* corresponding to this *scp-process*. Additionally, it should be noted that the sign of this structure itself and all meta-information about it are also included in this structure;
- if necessary, it is possible to manually unlock or lock some construction with some lock type using the corresponding *scp-operators* of the *scp-operator for lock control* class;
- after completing the execution of some *scp-process*, its text is usually deleted from *sc-memory*, and all blocked constructions are released (signs of structures that denoted locks are destroyed).

B. Concept of an scp-process

An ***scp-process*** is understood as some *action in sc-memory* that uniquely describes a specific act of executing some *scp-program* for given source data. If the *scp-program* describes an algorithm for solving a problem in a general way, then the *scp-process* denotes a specific action that implements this algorithm for the specified input parameters.

In fact, the ***scp-process*** is a unique copy created on the basis of the *scp-program*, in which each *sc-variable*, with the exception of *scp-variables'*, corresponds to the generated *sc-constant*.

Belonging of some action to a set of *scp-processes* guarantees the fact that only signs of elementary actions (*scp-operators*) will be present in the decomposition of this action, which can be interpreted by the implementation of an *Abstract scp-machine*.

C. Concept of an scp-operator

Each ***scp-operator*** represents some elementary *action in sc-memory*. The arguments of the *scp-operator* will be called operands. The order of the operands is specified using the appropriate role relations (*1'*, *2'*, *3'*, and so on). The operand marked with role relation *1'* will be called the first operand, marked with role relation *2'* – the second operand, etc. The type and meaning of each operand is also specified using various subclasses of the *scp-operand'* relation. In general, as the operand, any *sc-element* can act, including the sign of any *scp-program*, including the program itself containing this operator.

Each ***scp-operator*** must have one or more operands, as well as an indication of the ***scp-operator*** (or several) that should be executed next. The exception to this rule is the *scp-operator for program completion*, which does not contain a single operand and after which execution no *scp-operators* can be executed within this program.

Each ***atomic type of the scp-operator*** is a class of *scp-operators*, which is not divided into more particular ones and, accordingly, is interpreted by the implementation of the *Abstract scp-machine*.

Let us consider the upper level of the classification of scp-operators, which is given in more detail in [3].

scp-operator
⊂ *action in sc-memory*
⇐ *family of subsets**:
atomic type of the scp-operator
⇒ *subdividing**:
{• *scp-operator for generating constructions*
• *scp-operator for associative search of constructions*
• *scp-operator for deleting structures*
• *scp-operator for checking conditions*
• *scp-operator for controlling the values of operands*
• *scp-operator for controlling scp-processes*
• *scp-operator for event control*
• *scp-operator for processing files contents*
• *scp-operator for lock control*
}

The role relation ***initial operator'*** specifies those *scp-operators* that should be executed first within the decomposition of the *scp-process* that corresponds to the *scp-program*, i.e. those with which, actually, the execution of the *scp-process* begins.

D. Parameters of scp-programs

parameters of the scp-program'
⊂ *action argument'*
⇒ *subdividing**:
{• *in-parameter'*
• *out-parameter'*
}

The role relation ***parameter of the scp-program'*** links the sign of the *scp-process* with its arguments, that corresponds to the *scp-program*.

Parameters of the ***in-parameter'*** type, although they correspond to *variables of the scp-program'*, cannot change the value during its interpretation. The fixed value of the variable is set when creating a unique copy of the *scp-program* (*scp-process*) for its interpretation, and thus the corresponding *scp-variable'* at the time of its interpretation becomes an *scp-constant'* within each *scp-operator* in which this *scp-variable'* occurred. The usage of *in-parameters* can be considered by analogy with the usage of a variant of the value transfer mechanism in traditional programming languages, with the condition that the value of a local variable within a child program cannot be changed.

Parameters of the ***out-parameter'*** type correspond to *variables of the scp-program'* and have all the same corresponding properties. Most often, it is assumed that the value of this parameter is necessary for the maternal *scp-program* containing the call operator of the current *scp-program*. At the same time, at the moment of the beginning of interpretation, a node denoting a variable (or rather, its unique copy within the process) of the maternal process is passed directly to the child process as a parameter. The specified variable may, if necessary, have a value or not. After completion and during the interpretation of the child process, the maternal process can still work with the variable passed as the *out-parameter'*, viewing or changing its value if necessary. The usage of the out-parameter can be considered by analogy with the usage of the link transmission mechanism in traditional programming languages.

X. Model for the interpreter of the Basic programming language of ostis-systems

The advantages of the proposed multi-agent approach to building knowledge processing machines and, accordingly, problem solvers can work not only at the platform-independent level but also at lower levels. So, in particular, the interpreter of the *Basic programming language of ostis-systems* is also proposed to be built as a *non-atomic abstract sc-agent* that provides interpretation of the methods described in the *SCP Language*. Thus, such an interpreter is included in the general hierarchy of agents that build-up the *knowledge processing machine* of *ostis-systems* and is an *abstract sc-agent that is not implemented in the SCP Language*.

In general, there may be many options for implementing such interpreters. Within the *OSTIS Standard*, one of them is offered as a standard and is called an *Abstract scp-machine*.

Abstract scp-machine
∈ *abstract sc-agent that is not implemented in the SCP Language*
⇒ *decomposition of an abstract sc-agent**:
{• *Abstract sc-agent for creating scp-processes*
• *Abstract sc-agent for interpreting scp-operators*

- *Abstract sc-agent for synchronizing the process of interpreting scp-programs*
- *Abstract sc-agent for destroying scp-processes*
- *Abstract sc-event for synchronizing events in sc-memory and its implementation*
  - ⇒ *decomposition of an abstract sc-agent**:
    - {• *Abstract sc-agent for translating the generated event specification in sc-memory into an internal representation*
    - • *Abstract sc-agent for processing an event in sc-memory that initiates an agent scp-program*

    }

}

The purpose of an *Abstract sc-agent for creating scp-processes* is to create *scp-processes* corresponding to a given *scp-program*. This *sc-agent* is activated when an *initiated action* belonging to the *action of interpreting scp-program* class appears in *sc-memory*. After the *sc-agent* checks the initiation condition, the *scp-process* is created taking into account the specific parameters of the interpretation of the *scp-program*, after which the *initial operator'of the scp-process* is searched and added to the the set of *real entities*.

The purpose of the an *Abstract sc-agent for interpreting scp-operators* is actually the interpretation of the operators of the *scp-program*, that is, the execution in *sc-memory* of actions described by a specific *scp-operator*. This *sc-agent* is activated when an *scp-operator* belonging to the *real entities* class appears in *sc-memory*. After performing the action described by the *scp-operator*, the *scp-operator* is added to the set of *past entities*. In the case when the semantics of the action described by the *scp-operator* suggests the possibility of branching for the *scp-program* after executing this *scp-operator*, then one of the subsets of the class of *performed actions – unsuccessfully performed action* or *successfully performed action* is used.

The purpose of an *Abstract sc-agent for synchronizing the process of interpreting scp-programs* is to provide transitions between *scp-operators* within a single *scp-process*. This *sc-agent* is activated when some *scp-operator* is added to the set of *past entities*. Next, a transition is made along the *sc-arc* belonging to the *sequence of actions** relation (or more particular relations, if the *scp-operator* was added to the set of *successfully performed actions* or *unsuccessfully performed actions*). In this case, the next *scp-operator* becomes a *real entity* (active *scp-operator*) if at least one *scp-operator* associated with it by incoming *sc-arcs* belonging to the *sequence of actions** relation (or more particular relations) became a *past entity* (or, respectively, a subset of past entities). In the case when it is necessary to wait for the completion of all previous operators, the operator of the *conjunction of preceding operators* class is used for synchronization.

The purpose of an *Abstract sc-agent for destroying scp-processes* is the destruction of the *scp-process*, i.e. the deletion from *sc-memory* of all *sc-elements* that build it up. This *sc-agent* is activated when an *scp-process* belonging to a set of *past entities* appears in *sc-memory*. At the same time, the destroyed *scp-process* does not necessarily have to be fully formed. The need to destroy an incomplete *scp-process* may arise if, when creating the *scp-process*, problems arose that did not allow continuing the creation of the *scp-process* and its performance.

The purpose of an *Abstract sc-agent for event synchronization in sc-memory and its implementation* is to ensure the operation of *non-atomic sc-agents* implemented in the *SCP Language*.

The purpose of an ***Abstract sc-agent for translating the generated event specification in sc-memory into the internal representation*** is the translation of oriented pairs describing the *primary initiation condition** of some *sc-agent* into the internal representation of elementary events at the level of *sc-storage*, provided that this *sc-agent* is implemented at a platform-independent level (using the *SCP Language*). The condition for initiating this *sc-agent* is the appearance in *sc-memory* of a new element of the set of *active sc-agents*, for which the corresponding oriented pair will be found and translated.

The purpose of an *Abstract sc-agent for event processing in sc-memory, initiating the agent scp-program* is to search for an *agent scp-program*, included in the set of *sc-agent programs** for each *sc-agent*, the primary initiation condition of which corresponds to an event that occurred in *sc-memory*, as well as the generation and initiation of an action aimed at interpreting this program. As a result of the operation of this *sc-agent*, an *initiated action* appears in *sc-memory*, belonging to the *action* of *interpreting scp-program* class.

XI. Conclusion and directions for further development

In the article, the current problems in the field of developing hybrid problem solvers are formulated and an approach to solving some particular problems that are part of these more general problems is proposed. Thus, the solution of the formulated general problems is still relevant, however, the usage of the *OSTIS Technology* and the principles proposed in this work for constructing problem solvers based on it creates preconditions for their solution.

It is possible to formulate a number of more specific directions for the development of the approaches proposed in the article:

- Integrate ideas of situational control into the proposed approach more closely and fully;
- Refine the proposed locking mechanism, in particular, to minimize the number of lock classes, to take into account and implement the ideas of implementing lock-free algorithms;
- Eliminate the need to introduce sc-meta-agents and scp-meta-programs.
- Modify the *SCP Language* in order to be capable of describing the receptor and effector interaction of ostis-systems within scp-programs.
- When developing an Abstract scp-machine, to take into account the principles of building wave programming languages [18], [19] and the ideas of insertion programming and modeling [20], [21].

Acknowledgment

The author would like to thank the research group of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for its help in the work and valuable comments.

The work was carried out with the partial financial support of the BRFFR (BRFFR-RFFR No. F21RM-139).

References

[1] A. Kolesnikov, *Gibridnye intellektual'nye sistemy: Teoriya i tekhnologiya razrabotki [Hybrid intelligent systems: theory and technology of development]*, A. M. Yashin, Ed. SPb.: Izd-vo SPbGTU, 2001.

[2] D. Shunkevich, "Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems," in *Open semantic technologies for intelligent systems*, V. Golenkov, Ed. BSUIR, Minsk, 2018, pp. 119–132.

[3] V. Golenkov, N. Guliakina, and D. Shunkevich, *Otkrytaja tehnologija ontologicheskogo proektirovanija, proizvodstva i jekspluatacii semanticheski sovmestimyh gibridnyh intellektual'nyh komp'juternyh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems]*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[4] A. Narin'jani, "Ne-faktory: kratkoe vvedenie [non-factors: a brief introduction]," *Novosti iskusstvennogo intellekta [Artificial intelligence news]*, no. 2, pp. 52–63, 2004.

[5] V. Gorodetskii, V. Samoilov, and D. Trotskii, "Bazovaya ontologiya kollektivnogo povedeniya avtonomnykh agentov i ee rasshireniya [Basic ontology of autonomous agents collective behavior and its extension]," *Izvestiya RAN. Teoriya i sistemy upravleniya [Proceedings of the RAS. Theory and control systems]*, no. 5, pp. 102–121, 2015, (in Russian).

[6] M. Wooldridge, *An Introduction to MultiAgent Systems - Second Edition*. Wiley, 2009.

[7] V. Tarasov, *Ot mnogoagentnykh sistem k intellektual'nym organizatsiyam [From multi-agent systems to intelligent organizations]*. M.: Editorial URSS, 2002, (in Russian).

[8] L. Cao, "In-depth behavior understanding and use: The behavior informatics approach," *Information Sciences*, vol. 180, no. 17, pp. 3067–3085, Sep. 2010. [Online]. Available: https://doi.org/10.1016/j.ins.2010.03.025

[9] L. Cao, T. Joachims, C. Wang, E. Gaussier, J. Li, Y. Ou, D. Luo, R. Zafarani, H. Liu, G. Xu, Z. Wu, G. Pasi, Y. Zhang, X. Yang, H. Zha, E. Serra, and V. Subrahmanian, "Behavior informatics: A new perspective," *IEEE Intelligent Systems*, vol. 29, no. 4, pp. 62–80, Jul. 2014. [Online]. Available: https://doi.org/10.1109/mis.2014.60

[10] M. Pavel, H. B. Jimison, I. Korhonen, C. M. Gordon, and N. Saranummi, "Behavioral informatics and computational modeling in support of proactive health management and care," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 12, pp. 2763–2775, Dec. 2015. [Online]. Available: https://doi.org/10.1109/tbme.2015.2484286

[11] G. S. Al'tshuller, *Najti ideju: Vvedenie v TRIZ — teoriju reshenija izobretatel'skih zadach, 3-e izd. [Find an idea: An introduction to TRIZ - the theory of inventive problem solving, 3rd ed.]*. M.: Al'pina Pablisher, 2010.

[12] G. P. Shhedrovickij, *Shema mysledejatel'nosti – sistemno-strukturnoe stroenie, smysl i soderzhanie [Scheme of mental activity – system-structural structure, meaning and content]*. M.: Shk. kul't. pol., 1995.

[13] D. Pospelov, *Situacionnoe upravlenie. Teorija i praktika [Situational management. Theory and practice]*. M.: Nauka, 1986.

[14] D. Shunkevich, "Ontological approach to the development of hybrid problem solvers for intelligent computer systems," in *Open semantic technologies for intelligent systems*, V. Golenkov, Ed. BSUIR, Minsk, 2021, pp. 63–74.

[15] E. W. Dijkstra, *Cooperating Sequential Processes*. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 65–138.

[16] C. A. R. Hoare, "Communicating sequential processes," *Commun. ACM*, vol. 26, no. 1, p. 100–106, jan 1983. [Online]. Available: https://doi.org/10.1145/357980.358021

[17] B. Chatterjee, S. Peri, M. Sa, and K. Manogna, "Non-blocking dynamic unbounded graphs with worst-case amortized bounds." Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2022/15795/

[18] P. Sapaty, "Jazyk VOLNA-0 kak osnova navigacionnyh struktur dlja baz znanij na osnove semanticheskih setej [WAVE-0 language as a basis for navigational structures for knowledge bases based on semantic networks]," *Izv. AN SSSR. Tehn. kibernet. [Izv. Academy of Sciences of the USSR. Tech. cybernet.]*, no. 5, pp. 198–210, 1986.

[19] D. I. Moldovan and Y.-W. Tung, "SNAP: A VLSI architecture for artificial intelligence processing," *Journal of Parallel and Distributed Computing*, vol. 2, no. 2, pp. 109–131, 1985. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0743731585900310

[20] A. Letichevskij, J. Kapitonova, V. Volkov, V. Vyshemirskij, and A. Letichevskij (Jr.), "Insercionnoe programmirovanie [insertion programming]," *Kibernetika i sistemnyj analiz [Cybernetics and systems analysis]*, no. 1, pp. 19–32, 2003.

[21] A. Letichevskij, "Insercionnoe modelirovanie [insertion modeling]," *Upravljajushhie sistemy i mashiny [Control systems and machines]*, no. 6, pp. 3–14, 2012.

Гибридные решатели задач интеллектуальных компьютерных систем нового поколения

Шункевич Д.В.

В работе сформулированы актуальные проблемы текущего состояния технологий разработки гибридных решателей задач, предложен подход к их решению на основе Технологии OSTIS. Сформулированы принципы построения решателя задач как иерархической системы навыков, основанной на многоагентном подходе, приведены онтологии агентов и выполняемых ими действий. Сформулированы принципы синхронизации деятельности агентов, а также разработана онтология базового языка программирования для реализации программ агентов и модель интерпретатора такого языка.

Received 01.11.2022

Semantic theory of programs in next-generation intelligent computer systems

Nikita Zotov
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: nikita.zotov.belarus@gmail.com

***Abstract*—Despite the active development and usage of programming languages, currently, there is no general theory of programs on the basis of which it would be possible to design and develop applied systems. In this article, the unified ontology of programming languages and representation of programs in different programming languages is proposed. The work demonstrates the features of the representation of programs and key points of the process of their interpretation.**

***Keywords*—knowledge representation language, programming language, method representation language, software computer system, ontological approach, denotational and operational semantics of a language, procedural programming language, non-procedural programming language, ostis-systems language**

I. Introduction

For a long period of development of computer systems (c.s.), hardware restrictions on solving various problems have been practically removed. The remaining restrictions are assigned to the share of the software. First of all, these limitations are related to the current problems of software development:

- hardware complexity outstrips mankind's ability to build software c.s. using the potential capabilities of hardware;
- skills and technologies of software development lag behind the requirements for developing programs of next-generation software development;
- the ability to use existing programs is threatened by the poor quality of their development.

The key to solving these problems is a deep understanding and competent usage of existing programming languages as the main tool for the mass creation of next-generation software c.s.

This article focuses on achieving the following results:

- (1) set out the classical foundations, reflecting the accumulated world experience in the field of programming languages;
- (2) systematize the main results in this area and represent them in the form of a unified semantic theory of programs.

In this article, the problems of the current state in the field of programs and programming languages that can and should be used to develop next-generation intelligent c.s. are described in detail. It is dedicated to the basic concepts of the theory of programming languages, gives an overview of the areas for applying programming languages that are quite in demand by modern human society, describes in detail the forms and contents of criteria for evaluating the effectiveness of languages, considers ways of representing and interpreting programs of various programming languages.

II. Current state issues

In the modern era of information technologies development, there are a huge number of programming languages, each of which has its own important purpose in the field of software system design. Each language demonstrates not only its specifics but also has its own advantages and disadvantages. The variety of programming languages [1], [2] and solutions created on them is so great that it is very easy to get lost in a sea of information about all aspects of the application and design of programming languages. In addition, the main problem is not the number of existing solutions in the field of programming languages but the number of forms (!) in which specific programming languages are represented. So, declarative knowledge, i.e. knowledge that is, for example, a specification of some program, and procedural knowledge, i.e. knowledge that is programs belonging to some programming language, are represented in completely different ways, methods, and means.

In connection with the above, the following key problems in the field of programming languages can be distinguished:

1) Since the number of programming languages grows with the increase in the need for them [3], the need for describing these programming languages for further usage and design of applied systems also grows. This, in turn, requires a high level of quality in the specification of a particular language: both a description of the syntax and semantics of the constructions of this language, as well as a description of the means and methods for renovating tools that provide interpretation or translation of this

language. That is, with an increase in the number of programming languages, not only the variety of forms of knowledge representation (programming languages) grows but also the number of software systems based on various forms of knowledge representation [4].

2) A wide variety of forms of knowledge representation, as mentioned above, provides a wide range of possibilities for designing software c.s. on each of them. It turns out that in order to integrate several software systems implemented in different programming languages, it is necessary to make sure that the systems can communicate with each other in each of the languages in which they are implemented [5], [6]. Thus, the striving to use existing software components is hampered by the implementation of the components themselves [7], since in order to combine these components it is necessary to change their program code [8], [9]. The presence of a variety of forms makes it difficult to implement compatible interoperable c.s. [10].
3) As the complexity of the program code grows, the number of humans able to understand its meaning decreases. Modern developers create software c.s. without taking into account its full life cycle [11]. Systems must be constantly updated and improved with the development of the technologies on which it is based [12]. This should be ensured by good documentation of implementing the components of these systems – this reduces not only the need to raise new resources and personnel but also helps to reduce the reengineering of software c.s. [13], [9].
4) Full automation of designing software c.s. is impossible, since the modern languages in which they are designed do not have the property of reflexivity – systems cannot cognize and understand themselves [14], [15], [16] and develop almost completely on their own. Thus, the existing intelligent c.s. are not intelligent as such, since they do not have the properties they require [17].
5) The key to easy and deep mastering of a specific language as the main professional tool of a programmer is understanding the general principles of building and using programming languages [18], [19], described by their general theory. Until today, a general theory of programming languages still does not exist, which makes it difficult to develop, verify, and use new and existing programming languages. Without a general theory of programming languages, everyone can develop fundamentally general methods and tools in the way they want but not the way is required [10] - it is necessary to agree on terms and concepts and multiply the results by creating next-generation interoperable computer systems [20].
6) Achieving the maximum of services and means at a minimum of costs is possible only through a deep understanding of the principles of building programming languages due to the simplicity of means and methods of knowledge representation. The complex should be reduced to the simple and explained in simple terms, without creating an additional illusion of importance [8], [12], [21].

All these problems are related and are problems of the current state of development directions in the field of Artificial intelligence [19], [22].

So, to solve these problems, it is necessary to create comfortable conditions for the implementation of computer systems that are semantically compatible and interoperable with each other. In the context of programming languages, a general theory of designing programs for next-generation intelligent c.s. is required, which:

1) allows integrating existing solutions in the field of designing programs for computer systems without much effort and costs [23];
2) will combine knowledge representation forms of declarative and procedural types;
3) will have a wide range of tools not only for describing the syntax and semantics of existing programming languages but also for designing new analogues;
4) will be understandable not only to human but also to machine [4];
5) denotes the principles by which next-generation programming languages should be designed.

The design of such general theories, strictly speaking, must be approached with a high degree of importance. Designed c.s. should always be able to use the properties that they are drawn. In order for this theory to be used as a certain system of knowledge about how to design and use programming languages and programs in software c.s. and how to interpret their programs, it is necessary for this theory to be described by means and methods by which these software c.s. are designed. We are talking about the fact that the ontological approach [24], [4], [23], [25], [26] is a fundamentally important approach to the design of a general theory of programs.

To implement these ideas, it is necessary to study and integrate the experience gained in the field of programming languages. Therefore, the results of other researches in the field of designing the general theory of programming languages and programs will be considered below.

III. Existing ontologies of programming languages

For the most part, the ideas proposed in scientific papers on the study of programming languages are certainly in demand and useful for designing software c.s. Thus, the idea that programming languages and programs implemented on them should be organized into a common taxonomy of concepts is fundamental, since it provides the highest quality environment for the design and implementation of c.s. The general theory of programs is needed not only to describe terms and concepts as some kind of specification used to design software c.s. (that is also important) but also in order to determine the quality of programming languages and programs on such issues as: "Is this language a programming language", "Is this knowledge a program", "How efficient is this program", "What is the degree of intelligence of this software system", etc. These ideas are proposed and discussed in the works of Raymond Turner [18], [27].

Until today, there are a large number of analogues for ontologies of programming languages and programs. The examples can be found in [28], [29]. It is also worth noting the developed ontologies of programs [14], [18], [30], [31], [32], [33], in which, strictly and unambiguously, the system of concepts is defined in formal languages – languages of logic and languages for describing the grammars of formal languages. However, none of them is such a result that could be used in the design of software c.s. without significant problems. The developed ontologies concentrate only a brief description of interconnected concepts, but the general picture of how these ontologies can be used in specific problems is almost unseen.

Today, there are completely opposite judgments about the purpose of programs and programming languages [34], which contradict the formal foundations of Artificial Intelligence [35]. There are more and more works related to the rethinking of information processing [36]. Software c.s. should not only be understandable to a human but should understand themselves, their capabilities, intentions, actions, and purposes, and understand cybernetic systems that are similar to them. Only in this way humanity and the results of its activities in the form of some specific systems will be able to work together, complementing each other and multiplying their results [10].

Based on the represented works, it can be concluded that:

- the general theory of programs and programming languages, which could be involved in solving any applied problem, as well as representing and implementing computer system design tools, has not been developed yet;
- unification of the representation of the means for description and implementation according to these descriptions as the main argument for operating the semantic knowledge representation, for complete mutual understanding between computer systems is not considered at all;
- programs and combinations of these programs in the form of program c.s. are implemented in most cases on an individual basis and are poorly documented, which complicates their usage, integration with other programs and software c.s., testing, and improvement.

The key to solving all these problems is the general technology for designing next-generation computer systems, on the basis of which it is possible to build a general theory of programs (programming discipline) [37], which will be considered further.

IV. Suggested solution

Despite the vast variety of classical technologies used by mankind, there is no general solution that allows solving the problem in a complex. Therefore, at the moment, the described problems can be solved only with the help of a general and universal solution – the OSTIS Technology. The OSTIS Technology is based on a unified version of information encoding based on semantic networks with a basic set-theoretic interpretation, called an SC-code. The language of semantic knowledge representation is based on two formalisms of discrete mathematics: set theory – defines the semantics of the language – and graph theory – defines the syntax of the language [38], [39]. Any types and models of knowledge can be described using the SC-code [40].

For the convenience of knowledge representation, there are three external knowledge representation languages based on the SC-code: SCg-code, with the help of which knowledge is displayed in the form of graph structures understandable to the average user, SCs-code, in which knowledge is represented in the form of linear text, SCn-code for displaying sc-constructions as hypertext. This representation is close to natural, understandable to the average user [40].

The OSTIS Technology is suitable for solving the listed problem, since:

1) The Standard of the OSTIS Technology [40] already implements the basic tools necessary for the design and development of interoperable c.s., which are based on the semantic knowledge representation. This eliminates not only the need to create top-level ontologies, which should be used in the general theory of programs as the basis for describing the concepts of this theory, but also helps to design solutions consistent with other ontologies. As a result, a common coherent world picture is formed, which is (1) consistent, that is, agreed, (2) unambiguously interpreted, (3) universal, and, (4) most importantly, understandable to everyone.
2) The OSTIS Technology is designed by a single unified knowledge representation language called an SC-code. The meaning of programs and programming languages is understandable and unambiguous if and only if this meaning is described in one common language understandable to any cybernetic system. The meaning lies not in the syntax of the signs, but

in the configuration of the connections between them (!) [40], [41], [42].

3) The SC-code is syntactically minimal. The minimum number of signs is used to describe objects and connections between them. At the same time, the diversity of these connections is reduced to the diversity of sign constructions. All this is provided by representing information in the form of graph structures [43], [44], [45].
4) The SC-code is not just convenient for describing and designing some complex objects – it can be used to design and implement any knowledge representation languages, including programs, computer systems, and, in general, the real world.
5) Ontological [46], [26] and component [47] approaches to the design of any complex objects ensure the fulfillment of the main principles by which modern systems should be designed. What is implemented and can be used, must be reused everywhere [48], [49].

Thus, the solution to all described problems is the general theory of programs, interpreted as an ontology of the general system, implemented through the OSTIS Technology.

V. General description of designed subject domains and ontologies

The result of this work is a *Subject domain and ontology of methods* (Subject domain and ontology of programs), which can be used to set methods (programs), their syntax, denotational and operational semantics. The *Subject domain and ontology of methods* is a private subject domain in relation to the *Subject domain and ontology of information constructions and languages*. This means that it inherits all the properties of the concepts and relations studied in it.

Subject domain and ontology of information constructions and languages

⇒ private subject domain*:
- Subject domain and ontology of languages
  ⇒ private subject domain*:
  - Subject domain and ontology of natural languages
  - Subject domain and ontology of formal languages

Subject domain and ontology of formal languages

⇒ private subject domain*:
- Subject domain and ontology of knowledge representation languages
  ⇒ private subject domain*:
  - **Subject domain and ontology of methods**

Subject domain and ontology of methods

⇒ private subject domain*:
- Subject domain and ontology of methods of ostis-systems
  ⇒ private subject domain*:
  - Subject domain and ontology of procedural methods of ostis-systems

∋ maximum studied object class′:
- method

∋ non-maximum studied object class′:
- method representation language
- method class
- meta-method
- process
- variable
- constant
- operator
- method quality

∋ explored relation′:
- submethod*
- subprocess*
- method syntax*
- parameter’
- start operator’
- denotational semantics of the method*
- operational semantics of the method*
- method of the specified method representation language*

VI. Concept of a method (program)

Each theory must be conceptually consistent. Despite the fact that there are different interpretations for the concept of a programming language in the literature, there should be a universal one. To do this, instead of programming languages, we will further talk about method representation languages and instead of programs of these programming languages – about methods as sign constructions of method representation languages (m.r.l.). This decision is justified by the fact that usually the language acts as a tool for some kind of knowledge of a certain type, and the term of the programming language is degenerate, since it is worth talking not about languages in which something can be programmed but about languages in which knowledge of a certain type can be represented, in this case – knowledge of a procedural kind. The terms of the programming language and the program themselves will be considered as non-basic identifiers for the concepts of the methods and method representation language, respectively.

Formally, a *method* is a specification for solving a problem of some class [40], [50]. The specification of each class of problems includes a description of the "binding" of the method to the initial data of a particular problem solved with the help of this method.

method

:= [program]

:= [description of how any or almost any action belonging to the corresponding action class can be performed]

:= [method for solving the corresponding class of problems that provides a solution to any or most problems of the specified class]

:= [generalized specification for solving problems of the corresponding class]

:= [program for solving problems of the corresponding class, which can be either procedural or declarative (non-procedural)]

:= [knowledge of how to solve problems of the corresponding class]

⊂ *knowledge*

∈ *knowledge type*

:= [way]

⊃ *problem-solving model*

VII. Concept of a method class. General classification of methods

Sometimes, it may be appropriate to allocate a certain subset of methods (for example, a set of methods with which a certain problem is solved), then in this case for these methods it is possible to describe the requirements that they must fulfill. Such sets of methods are *method classes* of some m.r.l., which are associated with a particular *problem-solving model*. Methods can be either *procedural* or *non-procedural* [18].

method class

⇐ *family of subclasses**:
method

:= [set of methods for which the representation (specification) of these methods can be unified]

:= [set of various problem-solving methods that have a common language for representing these methods]

:= [set of methods for which the representation language of these methods is set]

∋ *procedural problem-solving method*

    ⊃ *algorithmic problem-solving method*

∋ *non-procedural problem-solving method*

    ⊃ *logical problem-solving method*

    ⊃ *production problem-solving method*

    ⊃ *functional problem-solving method*

        ⊃ *artificial neural network*

    ⊃ *genetic "algorithm"*

:= [set of methods, which is associated with a particular problem-solving model]

Since each method corresponds to a generalized formulation of the problems solved using this method, each method class must correspond not only to a certain m.r.l. belonging to the specified *method class* but also to a specific language for representing generalized formulations of problems for different classes of problems, solved by methods belonging to the specified method class.

For procedural and non-procedural methods, although it is possible to set *input* and *output parameters*, the general denotational semantics of their logical elements cannot be set: for procedural methods, these are operators, for non-procedural methods – mathematical objects of the subject domain.

VIII. Concept of method representation language (programming language)

Each specific method class corresponds one-to-one to the m.r.l. belonging to this (specified) method class. Thus, the specification of each method class is reduced to the specification of the corresponding m.r.l., that is, to the description of its syntactic, denotational, and operational semantics. Examples of m.r.l. are all programming languages that basically belong to the subclass of m.r.l., but now the need to create effective formal m.r.l. for performing actions in the external environment of cybernetic systems is becoming increasingly important. Without this, complex automation [51], in particular, in the industrial sector, is impossible.

By *method representation language* we mean a formal language, (1) the sign constructions of which are the corresponding methods for which there are general building rules and (2) general rules for correlating with those entities and relations between them that are described by these methods.

With the help of m.r.l., *messages* (methods) for the computer are generated. These messages must be understandable (semantically correct and consistent) to the computer [52].

method representation language

:= [programming language]

⊂ *knowledge representation language*

    ⊂ *formal language*

:= [computer language]

:= [formal language, (1) the symbolic constructions of which are the corresponding methods for which there are general building rules and (2) general rules for correlating with those entities and relations between them that are described by these methods]

:=

[mean of communication between a human (user) and a computer (performer)]
:= [tool for producing software services]

A method belongs to a method representation language if it is a syntactically correct, syntactically consistent, semantically correct, and semantically consistent method of the specified m.r.l. (!).

relation set in multiple method representation languages^
:= [relation whose scope of definition includes many different method representation languages]
∋ *method of the specified method representation language**
∋ *syntactically correct method for the specified method representation language**
  := [method that does not contain syntax errors for the specified method representation language*]
  ⊂ *syntactically correct sign construction for the specified language**
∋ *syntactically consistent method for the specified method representation language**
  ⊂ *syntactically consistent sign construction for the specified language**
∋ *semantically correct method for the specified method representation language**
  := [method that does not contain semantic errors for the specified method representation language*]
  ⊂ *semantically correct sign construction for the specified language**
∋ *semantically consistent method for the specified method representation language**
  ⊂ *semantically consistent sign construction for the specified language**
  := [method of the specified method representation language that contains sufficient information to determine its truth*]

method of the specified method representation language*
:= [method belonging to the specified programming language*]
⊂ *text of the specified language**
⇒ *second domain**:
  method
⇐ *combination**:
  {• {}
    ⇐ *combination**:
      {• *syntactically correct method for the specified method representation language**
       • *syntactically consistent method for the specified method representation language**
      }
   • {}
    ⇐ *combination**:
      {• *semantically correct method for the specified method representation language**
       • *syntactically consistent method for the specified method representation language**
      }
  }

IX. General classification of method representation languages

In the modern information society, method representation languages (m.r.l.) are distinguished by their paradigms: *procedural*, *functional*, *logical*, *object-oriented* m.r.l., etc. The solution of the problem by the computer is made in the form of a sequence of operators: in the methods of functional m.r.l. – indication of other methods; in logical m.r.l., operators are used; and in object-oriented ones – objects.

method representation language
⊃ *general-purpose method representation language*
  := [general-purpose programming language]
⊃ *subject-oriented method representation language*
  := [subject-oriented programming language]
⇒ *subdividing**:
  *method representation language paradigm^*
  = {• *procedural method representation language*
     • *non-procedural method representation language*
    }

Procedural method representation languages set computations as a sequence of operators (commands). They are focused on computers with von Neumann architecture. Basic concepts of procedural m.r.l. closely related to computer components:

- variables of various types that model computer memory cells;
- assignment operators that model data transfers between memory areas;
- repetitions of actions in the form of iteration, which simulate the storage of information in adjacent memory cells;
- and more.

procedural method representation language
:= [imperative method representation language]
⊃ *structural method representation language*
  ∋ *example*′:
    • *Fortran*
    • *C*
    • *Pascal*
⊃ *object-oriented method representation language*
  ∋ *example*′:
    • *Smalltalk*
    • *Java*
    • *HTML*
  ⊃ *aspect-oriented method representation language*
⊃ *script method representation language*
  := [patch method representation language]

Non-procedural method representation languages, in contrast to procedural languages, set computations as a sequence of interconnected objects. Basic concepts of non-procedural m.r.l. usually are not related to computer components.

non-procedural method representation language
:= [declarative method representation language]
⊃ *logical method representation language*
  ∋ *example*′:
    • *Prolog*
⊃ *production method representation language*
⊃ *functional method representation language*
  := [applicative method representation language]
  ∋ *example*′:
    • *LISP*

X. Representation of the syntax and semantics of various methods

The *syntax* and *semantics* of a method represent its *specification*. The semantics of a method can be viewed from two perspectives: as a set of interrelated knowledge, which is determined by the denotational semantics of this method, and as knowledge that can be interpreted by another method, which is determined by the operational semantics of this method.

method specification*
⇒ *subdividing**:
  {• *method syntax**
  • *denotational semantics of the method**
    := [generalized formulation of the class of problems solved using this method*]
    ⇔ *semantically close sign**:
      *generalized formulation of the problems of the corresponding method class**
  • *operational semantics of the method**
    := [list of generalized agents providing method interpretation*]
    := [family of methods for interpreting this method*]
    := [formal description of the specified method interpreter*]
  }

A. Representing the syntax of the problem-solving method

Any method consists of atomic information constructions that set the order of actions in the knowledge base, with the help of which it is required to move from the initial state to the target one, thus solving some specific problem. So, for example, in a procedural method, any such operator represents some mathematical function. Expressions and operators are used to compose these functions into larger fragments. In turn, linear sequences of operators and conditional branches can also be represented by functions composed of functions inherent in particular components of these constructions. A cycle is easily described by a recursive function composed of the components included in its body.

The *method syntax** defines the set of its allowed constructions. The appearance of method elements is specified using a certain syntax. It describes such lexical details as the location of keywords and punctuation marks. Grammars are used to specify a particular syntax.

The syntax of m.r.l. in ostis-systems can be formally described in various ways. So, for example, it is possible to use the Backus-Naur meta-language to describe the syntax of some methods of a particular m.r.l. Other equally well-known forms of method representation are context-free grammars, extended Backus-Naur form, syntactic graphs [1], [53], [54].

However, it is much more logical and advisable to describe the syntax of other languages in the universal knowledge representation language – the *SC-code*. This approach will allow ostis-systems to independently understand, analyze, and generate texts of these languages on the basis of principles common to any form of external information representation, including non-linear ones [45]. Thus, languages written in the SC-code have the same syntax as the SC-code.

B. Representing the denotational semantics of the method

The semantics of a method explains the meaning of the syntactic constructions of a method. The most common methods for describing the semantics of programming languages are: denotational, operational, axiomatic, algebraic ones [55], [56]. Based on the principles of the OSTIS Technology, by the semantics of a method we mean the combination of the denotational and operational semantics of the method.

The description of how to "bind" a method to some class of problems includes:

- a set of variables that are included both in the method and in the generalized formulation of the problems of the corresponding class and whose values are the corresponding elements of the initial data of each specific problem being solved;
- part of the generalized formulation of problems of the class to which the method under consideration corresponds, which are a description of the conditions for applying this method;
- a description of the method initiation condition and its result;
- a description of initial and target situations in sc-memory.

"Binding" a method to a specific problem solved with the help of this method is carried out by searching for such a fragment in the knowledge base, that satisfies the conditions for applying the specified method. One of the results of such a search is the setting of a correspondence between the above variables of the method used and the values of these variables within a specific problem being solved. Another option for setting the correspondence under consideration is an explicit call of the corresponding method (program) with an explicit transfer of the corresponding parameters. However, this is not always possible, since when executing the process of solving a specific problem based on the declarative specification for performing this action, it is not possible to identify:

- when it is necessary to initiate a call (usage) of the required method;
- which specific method to use;
- which parameters, corresponding to the particular problem being initiated, must be passed in order to “bind” the method used to this problem.

A *process* is understood as some action in sc-memory that unambiguously describes a specific act of executing a certain method for given initial data [37]. If a method describes an algorithm for solving a problem in general terms, then a process denotes a specific action that implements this algorithm for given input parameters. In fact, the process is a unique copy created on the basis of a method in which each sc-variable corresponds to a generated sc-constant.

relation defined on a set (process)^
:= [relation whose scope of definition includes many possible processes]
∋ *parameter’*
⇒ *subdividing**:
{• *in-parameter’*
• *out-parameter’*
}
∋ *in-parameter’*
∋ *out-parameter’*
∋ *initial information construction’*
∋ *subprocess**

The process of “binding” a problem-solving method to a specific problem solved using this method can also be represented as a process consisting of the following phases:

- building a copy of the used method;
- pasting the main (key) variables of the method used together with the main parameters of a specific problem being solved.

As a result, on the basis of the considered method used as a sample (template), a specification of the process for solving a specific problem is built. The description of the process of “binding” the solution method to a specific problem, as well as the description of the elements of the method, is the *denotational semantics of this method*.

denotational semantics of the method
∋ *general formulation of the class of problems**
:= [text formulation of the set of problems solved by this method]
⊂ *explanation**
∋ *primary initiation condition**
∋ *initiation condition and result**
⇐ *Cartesian product**:
⟨• *method class*
• *implication**
⟩
∋ *condition of initial and target situations**
⇐ *Cartesian product**:
⟨• *method class*
• *implication**
⟩

An example of the part of the specification that describes the denotational semantics of the Method for finding the double sum of two numbers is demonstrated in Figure 1.

The *general formulation of the class of problems** relation is a class of sc-connectives between an sc-connective, denoting a set of methods, and an ostis-system file, which is an explanation of which classes of problems can be solved using a given set of methods. In some rare cases, the presence of such an sc-connective may not be in the specification of a method, since there is no need to specify which classes of problems can be solved using this method.

The connectives of the *primary initiation condition** relation connect the sc-connective, denoting a set of methods, and the binary oriented pair, describing the primary condition for initiating a given method, i.e. such a

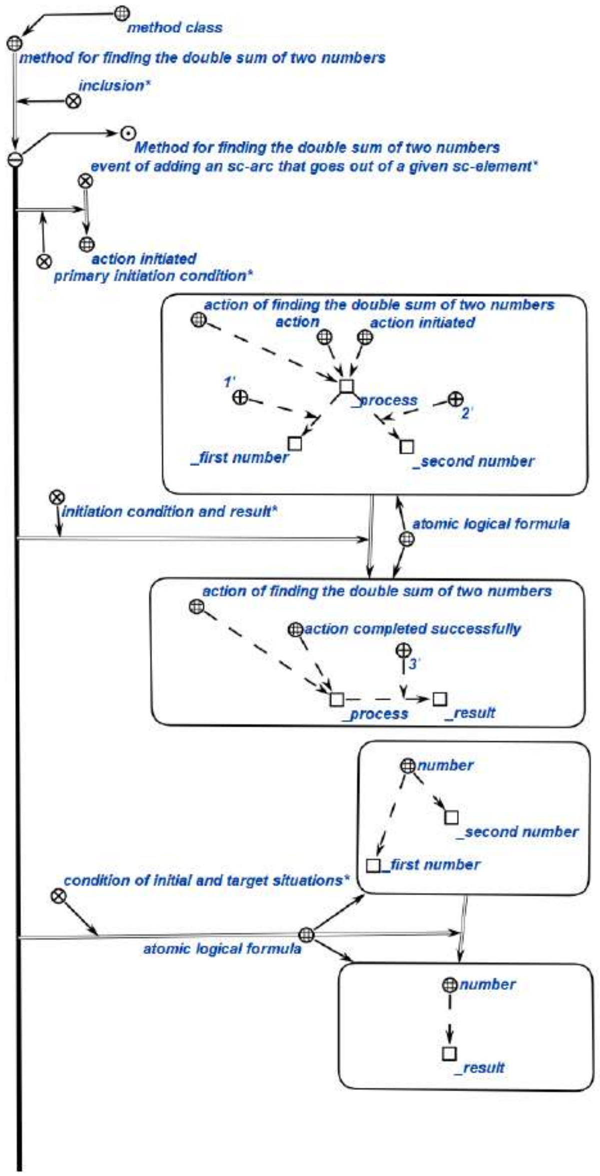

method class
method for finding the double sum of two numbers
inclusion*
Method for finding the double sum of two numbers
event of adding an sc-arc that goes out of a given sc-element*
action initiated
primary initiation condition*
action of finding the double sum of two numbers
action
action initiated
1'
_process
2'
_first number
_second number
initiation condition and result*
atomic logical formula
action of finding the double sum of two numbers
action completed successfully
3'
_process
_result
number
_second number
_first number
condition of initial and target situations*
atomic logical formula
number
_result

Figure 1. The specification of a method for solving the problem of calculating the double sum of two numbers

specification of the situation in sc-memory, the occurrence of which prompts the meta-method-executor to transfer the given set of methods into the active state and begin checking for their full initiation condition.

The first component of this oriented pair is the sign of some class of elementary events in sc-memory*, for example, the event of adding an sc-arc going out of a given sc-element*.

In the general case, the second component of this oriented pair is a random sc-element, with which the specified type of event in sc-memory is directly associated, i.e., for example, the sc-element, from which the generated or deleted sc-arc or file, the contents of which have been changed, goes out, or in which this sc-arc or the file come.

The connectives of the initiation condition and result* relation link together the sc-connective, denoting the set of methods, and a binary oriented pair, linking the initiation condition for this set of methods and the results of executing this set of methods in any particular system. The specified oriented pair can be considered as a logical implication connective, while the universality quantifier is implicitly imposed on sc-variables present in both parts of the connective and the existence quantifier is implicitly imposed on sc-variables present either only in the premise or only in the conclusion.

The first component of the specified oriented pair is a logical formula that describes the condition for initiating the described method, that is, the construction, the presence of which in sc-memory calls a lot of methods to start working on changing the state in sc-memory. This logical formula can be both atomic and non-atomic, which allows using any connectives of the logical language.

The second component of the specified oriented pair is a logical formula that describes the possible results of performing the described set of methods, that is, a description of the changes in the state of sc-memory made by it. This logical formula can be both atomic and non-atomic, which allows using any connectives of the logical language.

The connectives of the *condition of initial and target situation** relation connect an sc-connective, denoting a set of methods, and a binary oriented pair, connecting the initial and target situations in sc-memory, that is, in short, the situation before applying the method and the desired situation after applying the method. The specified oriented pair can also be considered as a logical implication connective, while on the sc-variables present in both parts of the connective the universal quantifier is implicitly imposed, and on the sc-variables present either only in the premise or only in the conclusion the existential quantifier is implicitly imposed. For the first and second components of the specified oriented pair, the same restrictions and properties are imposed as for the components of the oriented pair, which is the second component of the *initiation condition and result** relation.

It should be noted that the connectives of the *initiation condition and result** relation and the *condition of the initial and target situation** relation can be represented differently. Sometimes, it may not be necessary to create and check the second condition of the method, which checks for the presence of the initial situation in sc-memory and checks for reaching the target situation in sc-memory as a result of applying the method. If so, then the condition of the initial and target situation* can be specified in the logical formulas that are components in the second component of the connective of the *initiation condition and result** relation.

Programs, depending on the way of their representation in languages, will differ. This can be verified by comparing examples of procedural (Fig. 2) and logical (Fig. 3) methods for solving the same problem.

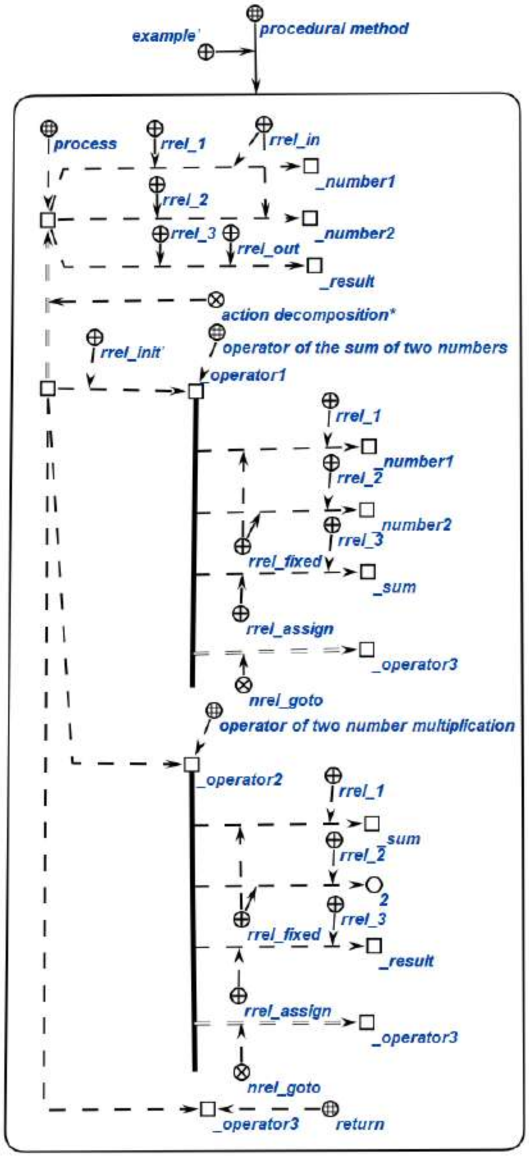


Figure 2. An example of a procedural method for solving the problem of calculating the double sum of two numbers

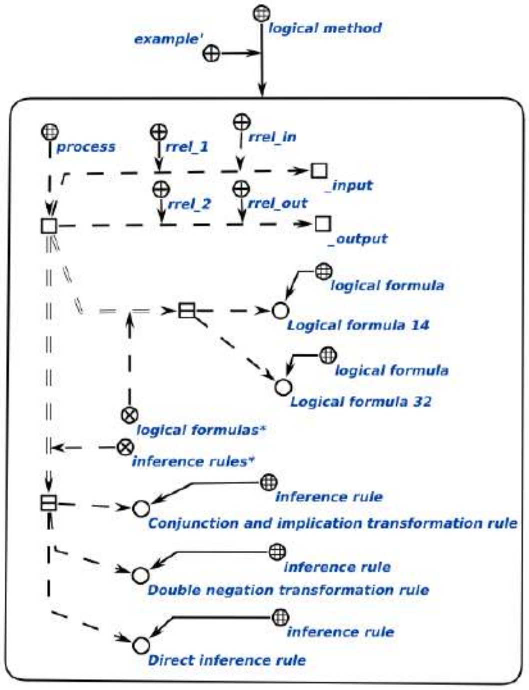


Figure 3. An example of a logical method for solving the problem of calculating the double sum of two numbers

With the help of the SC-code, it is also possible to represent those languages that are not written in it. The problem will be in the fact that the form and meaning of the language and its methods will be separated, that is, they will be represented in different ways. In this case, the SC-code is a powerful tool for integrating the specifications of various languages of external knowledge representation. However, it should be noted that there is no need to represent different forms of methods belonging to different method representation languages within the OSTIS Technology. This is explained by the following facts:

1) The SC-code is a fairly universal language for representing any kind of knowledge. This means that different forms of the algorithm for solving the same problem can be minimized. In the SC-code, the foundation is a formal theory, which provides a universal representation of various types of declarative and procedural knowledge. Thus, logical methods can be represented as procedural programs, in which as operands of operators not only logical formulas and inference rules will serve but also other methods that provide interpretation of these logical formulas using inference rules. Thus, the SC-code can be called not only a language of unified knowledge representation but also a language in which different classes of problems can be solved in the same way.
2) Various types of knowledge in ostis-systems, designed according to the principles of the OSTIS Technology, are deeply integrated with each other. This provides not only simplicity for creating these systems based on existing languages that can be described in the SC-code but also great opportunities for creating basic programming languages for next-generation computer systems, such as, for example, the basic language for representing SCP procedural methods, the basic language for representing production methods, etc. Modern method representation languages are created to simplify the description of

some algorithm for fast and high-quality solution of a certain class of problem [57]. In turn, the proposed methods and models make it possible to design an m.r.l. for next-generation computer systems with the help of basic knowledge representation languages in such a way that the very form of knowledge representation does not change. Methods of different m.r.l. must have one universal form of representation, i.e. the same syntax, but may allow the denotational and operational semantics of their methods to be described and represented in different ways using the same syntax.

3) Designing new m.r.l. should be reduced to their full description in the minimum family of SC-code languages: the SC-code itself, SCP, and SCL. We are talking about designing a new method representation language: it is enough to develop a (non-atomic) meta-method in SCP and SCL languages, which will interpret the methods of the languages being designed and also describe the denotational semantics of these methods. Meta-method for interpreting m.r.l. methods can be called an interpreter of these languages, that is, some abstract sc-machine on which it is possible to execute methods of a certain language for representing these methods.

C. Representing the operational semantics of the method

A complete *method specification**, in addition to the *denotational semantics of this method**, must include *the operational semantics of this method**, that is, a formal description of the interpreter of the given method. Operational semantics of the m.r.l. describes the execution of a method written in a given language by means of a virtual computer. A virtual computer is defined as an abstract automaton. The internal states of this automaton model the states of the computational process when the method is executed. The automaton translates the source text of the method into a set of formally defined operations. This set defines the transitions of the automaton from the initial state to the sequence of intermediate states by changing the values of the method variables. The automaton completes its work by passing to some final state. Thus, here we are talking about a fairly direct abstraction of the possible usage of m.r.l. Operational semantics describes the meaning of a method by executing its operators on a simple automaton. The changes that occur in the state of the machine, when a given operator is executed, determine the meaning of that operator.

The operational semantics of a specific method is reduced to the description of a *meta-method* that interprets it, verifies, etc.

meta-method
⊂ *method*
:= [method whose parameter values are other methods]

operational semantics of the method
∋ *interpretation meta-method**
  ⇐ *Cartesian product**:
    ⟨• *method class*
     • *method*
    ⟩
∋ *meta-method for verification and quality assessment**
  ⇐ *Cartesian product**:
    ⟨• *method class*
     • *method*
    ⟩

The *interpretation meta-method** relation is a class of sc-connectives between an sc-connective, denoting a set of methods, and an sc-node, denoting a method that is capable of interpreting a given set of methods. The *meta-method of verification and quality assessment** is a class of sc-connectives between an sc-connective, denoting a set of methods, and an sc-node, denoting a method that is capable of verifying and evaluating the quality of a given set of methods.

Within the OSTIS Technology, there can be a wide variety of such meta-methods. Each of them can consist of many atomic and non-atomic submethods. These can be both meta-methods that interpret the methods of certain m.r.l. and meta-methods that verify and analyze the quality of these methods. In addition, meta-methods can perform operations on other meta-methods.

meta-method for methods interpreting base method representation languages
⇒ *inclusion**:
- *meta-method for methods interpreting the SCP procedural method representation language*
- *meta-method for methods interpreting the SCL logical method representation language*
- *meta-method for methods interpreting the production method representation language*
- *meta-method for methods interpreting the functional method representation language*
- *meta-method for methods interpreting the neural network representation language*
- *meta-method for methods interpreting the representation language of genetic algorithms*

meta-method for verifying and evaluating the quality of methods in basic method representation languages

⇒ *inclusion**:
- *meta-method for verifying and evaluating the quality of methods in the SCP procedural methods representation language*
- *meta-method for verifying and evaluating the quality of methods in the representation language of logical SCL methods*
- *meta-method for verifying and evaluating the quality of methods in the representation language of production methods*
- *meta-method for verifying and evaluating the quality of methods in the representation language of functional methods*
- *meta-method for verifying and evaluating the quality of neural network representation language methods*
- *meta-method for verifying and evaluating the quality of methods for the representation language of genetic algorithms*

The concepts of syntax, denotational and operational semantics of method representation languages are reduced to the concepts of syntax, denotational and operational semantics of any language in general.

XI. Representation of the syntax and semantics of method representation languages

It is clear that in order to use the m.r.l., each language construction should be described separately, as well as its usage in aggregate with other constructions. There are many different constructions in a language, the exact definition of which is necessary both for the programmer using the language and for the developer of the compiler for that language. This knowledge allows the programmer to predict the calculations performed by the method operators. The constructions descriptions are necessary for the developer to create a correct implementation of the compiler.

A description of a formal model of a method representation language can be given by its *specification*. The specification contains a description of the syntax and semantics of the m.r.l.

method representation language specification*

⊃ *relation posed on a set (method representation language)**

⇒ *subdividing**:

{• *syntax of the method representation language**
 ⊂ *language syntax**
 := [be a theory of well-formed information constructions belonging to a given method representation language]
• *denotational semantics of the method representation language**
 ⊂ *language denotational semantics**
 := [generalized formulation of the classes of problems solved using this method representation language*]
• *operational semantics of the method representation language**
 ⊂ *language operational semantics**
 := [list of generalized agents that provide interpretation of methods of a given method representation language*]
 := [family of methods for interpreting texts in a given method representation language*]
 := [formal description of the interpreter of the specified method representation language*]
}

The *syntax of m.r.l.** is a binary oriented relation, each pair of which associates a sign of some language with a description of syntactically allocated classes from fragments of constructions of a given m.r.l. with a description of relations defined on these classes and with conjunction of quantifier propositions, which are the syntactic rules of the given language, that is, the rules that all syntactically correct (well-built) constructions of the specified m.r.l. must satisfy. In the general case, the *syntax of the m.r.l.** relation is no different from the *language syntax** relation, but still there is a refinement, since m.r.l. are languages in general, and the syntax of the m.r.l. inherits all syntax properties of any languages. The *syntax of the m.r.l** combines the syntaxes of all methods belonging to a given method representation language.

*Denotational semantics of m.r.l.** means a binary oriented relation, each pair of which associates a sign of some language with the sign of some ontology, which can be used to describe the methods of this language, and *operational semantics of m.r.l** is a description of the meta-method for interpreting the methods of this language.

In the context of this work, specific types of denotational and operational semantics will not be considered further.

XII. Help-system for design and method development support

The current state of the art in software design and development suggests that developers are more eager to automate the development of methods in specific method representation languages than to provide training tools for their design, including the design of new method representation languages. This leads to the following problems:

1) While the number of developers who understand the code of a complex software system is decreasing, the requirements for that system are growing faster and faster. Often, developers of complex software systems themselves are not able to explain the logic of these systems. For this reason, it is necessary to create tools that will automate the documentation of software systems [52].
2) To train new developers in the skills of working with software systems and their development, it is necessary to attract the resources of development experts who understand the principles of operation of these software systems. The problem is solved by developing a help system that will not only teach the user how to design problem solving methods and software systems based on these methods, but also point out gaps in related disciplines necessary to achieve high-quality results of all their activities.
3) In engineering, developers often design and develop solutions that have already been created by other specialists. Thus, functionally equivalent methods of solving problems are obtained, and even software systems that solve similar problems. The key to solving this problem is to design a semantically powerful library of reusable problem solving methods.

Thus, the semantic theory of programs alone is not enough. In addition to it, for a permanent and unhindered design and development of methods of a different class, it is necessary to develop:

1) an intelligent help system for supporting the design and development of methods, mentioned in [58], which will not only help the developer verify the method being developed, but also suggest ways to develop it;
2) a semantically powerful library of reusable components [47] for quickly finding existing problem solving methods and applying them to other more complex problems [46].

The potential help system should be part of a common development tool for next-generation intelligent computer systems - ostis-platform [59] - and may consist of the following components:

- the intelligent help-system on the semantic theory of programs;
- the intelligent help system on the library of reusable problem solving methods,
- the intelligent help-system for a set of tools for designing methods for solving problems,
- the intelligent help-system on the methodology of teaching the design of various methods for solving problems.

Each component contains knowledge from the relevant area of design and development theory of problem solving methods. In accordance with open semantic technology, each component must include:

- reference subsystem,
- subsystem for monitoring and analyzing the activities of the developer of methods for solving problems,
- learning management subsystem.

Each of the subsystems interacts with other subsystems and can also function autonomously.

The reference subsystem is an expert consultant in the field of semantic program theory who can answer any question from a novice or experienced user. Each of these systems can become individual assistants in the training of new specialists - a personal ostis-assistant. The functions of the reference subsystem include:

- search for information at the request of the user, including freely-designed ones;
- displaying the information found, taking into account the user's skill level;
- analysis of program texts and making suggestions to improve their effectiveness;
- generation of program texts on request to the user;
- self-initiation in case of difficulties for the user or the student.

Thus, the development of such components according to the principles of the OSTIS Technology will confirm the general semantic theory of programs.

XIII. Quality (efficiency) criteria of methods

The method representation language can be defined by a set of indicators that characterize its individual properties. The problem arises of introducing a measure to assess the degree of suitability of the m.r.l. to the performance of the functions assigned to it – *method quality* [6], [56], [60]. The quality criteria of methods are given on the basis of particular indicators of the efficiency of these methods (quality indicators). The method of connection between particular indicators determines the type of efficiency criterion.

method quality

⇒ *prerequisite property**:

- *ease of reading and understanding the method*
- *ease of creating the method*
- *method cost*
- *total volume of problems solved using this method class*

- *variety of types of problems solved using this method class*
- *method reliability*

Ease of reading and understanding the method should make it easy to highlight the basic concepts of each part of the method without referring to its specification.

ease of reading and understanding the method

⇒ *prerequisite property**:
  - *m.r.l. syntax simplicity*
  - *orthogonality of m.r.l. information structures*
  - *structured flow of control in a method*

The method representation language should provide a *simple* set of informational constructions that can be used as basic elements when creating methods. The syntax of the language has a strong impact on simplicity: it must transparently reflect semantics of constructions, exclude ambiguity and non-uniqueness of interpretation.

Orthogonality means that any possible combination of different information constructions will be meaningful, with no unexpected behavior resulting from the interaction of the constructions or context of usage.

The order of control transfers between method operators, i.e. the *flow of control*, should be human readable and understandable.

Ease of creating the method reflects the convenience of the language for representing that method in a particular subject domain.

ease of creating the method

⇒ *prerequisite property**:
  - *m.r.l. syntax simplicity*
  - *m.r.l. natural syntax*
  - *orthogonality of m.r.l. information structures*
  - *completeness and accuracy of m.r.l. specification*
  - *consistency and integrity of m.r.l. specification*

The syntax of the method should facilitate an easy and transparent display of the algorithmic structures of the subject domain in it. The syntax of m.r.l. should be not only *simple*, but also *natural*, and support the *orthogonality* of language informational constructions.

Ease of representation of a new method is ensured by *complete and precise, consistent and integral specification* of the appropriate language. That is, it is required to have a sufficient number of information constructions in this language in order to represent a particular method. At the same time, the language specification must be consistent and integral in order to represent consistent methods.

Cost of the m.r.l. method is made up of several components.

method cost

⇒ *prerequisite property**:
  - *cost of method applying*
  - *cost of method interpretation*
  - *cost of method creating, testing, and using*
  - *cost of method maintenance*

Cost of method applying largely depends on the structure of the m.r.l. A language that requires numerous syntactic type checks during method application will prevent the program from running quickly.

Cost of method interpretation depends on the capabilities of the interpretation meta-method used. The more perfect the optimization methods are, the more expensive will be the interpretation costs. The amount of the cost of creating, testing, and using the method depends on the used meta-method of verification and evaluation of the quality of this method.

Numerous studies show that a significant part of the cost of the method used is not the cost of its development but the *cost of its maintenance* [11]. Associating method maintenance with other method characteristics, the dependence on readability, since maintenance usually occurs by the next generation of developers, should first of all be highlighted.

The total volume of problems and the variety of types of problems solved with the help of this method class are no less important characteristics, which show the degree of universality of the corresponding m.r.l. The more problems can be solved on m.r.l., the more universal it is.

Reliability of m.r.l. methods should be ensured by a minimum of errors during the operation of a particular method.

All of these criteria can be applied to the method representation languages themselves.

XIV. Directions of development

This article is the beginning of the semantic theory of programs for next-generation c.s. The logical development of this work will be:

- refinement and addition of concepts of the *Subject domain and ontology of methods* to achieve the completeness of the theory;
- description of private subject domains of the *Subject domain and ontology of methods* for specific types of methods, as well as clarification of the denotational and operational semantics of the specification of these methods;
- description of possible ways of implementing meta-methods for interpreting methods of various m.r.l;
- implementation of tools to support the design and development of various methods for solving the

problem and the development of their respective specifications;
- formalization of mathematical models for calculating method efficiency estimates.

XV. Conclusion

The main conclusion of this work is that it is necessary not to replenish knowledge about which programming languages already exist and to reveal possible areas of their application, but to develop fundamentally new programming languages with which it was possible to create next-generation intelligent computer systems with high level of intelligence, semantic compatibility and interoperability with similar computer systems, unification of knowledge representation and processing, platform independence from tools for their implementation, and so on.

Such systems should be developed according to the principles of the OSTIS Technology, and their main development languages will be graph languages for representing methods that are sublanguages in relation to the basic procedural programming language SCP.

In this article, the problems of ensuring the design of software systems are considered. A comparative analysis of existing solutions in the field of unifying the representation of programming languages has been carried out. The work defines the solution of the problem in the form of designing and developing a universal theory of programming languages according to the principles underlying the OSTIS Technology. This article is also a specification of how software systems should be specified and designed.

Acknowledgment

The author would like to thank the research groups of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics and the Brest State Technical University for their help in the work and valuable comments.

References

[1] Sebesta, R. W, *Concepts of Programming Languages*. 10th ed. — Pearson/Addison-Wesley, 2012.

[2] Tourlakis, George, *Computability*. Springer Nature, 2022.

[3] A. Iliadis, "The tower of babel problem: making data make sense with basic formal ontology," *Online Information Review*, vol. 43, no. 6, pp. 1021–1045, 2019.

[4] C. M. Zapata Jaramillo, G. L. Giraldo, and G. A. Urrego Giraldo, "Ontologies in software engineering: approaching two great knowledge areas," *Revista Ingenierías Universidad de Medellín*, vol. 9, no. 16, pp. 91–99, 2010.

[5] Golenkov, V., Guliakina, N., Davydenko, I., Eremeev, A., "Methods and tools for ensuring compatibility of computer systems," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2019, pp. 25–52.

[6] Robert Martin, *Clean code. Creation, analysis and refactoring*, 2021.

[7] Ryndin, Nikita and Sapegin, Sergey, "Component design of the complex software systems, based on solutions' multivariant synthesis," *International Journal of Engineering Trends and Technology*, vol. 69, pp. 280–286, 12 2021.

[8] D. Posnett, A. Hindle, and P. Devanbu, "A simpler model of software readability," in *Proceedings of the 8th working conference on mining software repositories*, 2011, pp. 73–82.

[9] S. Scalabrino, M. Linares-Vasquez, D. Poshyvanyk, and R. Oliveto, "Improving code readability models with textual features," in *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*. IEEE, 2016, pp. 1–10.

[10] Gulyakina N. A., Golenkov V. V., "Graphic-dynamic models of parallel knowledge processing: principles of construction, implementation and design," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, Golenkov V. V., Ed. BSUIR, Minsk, 2012, pp. 23–52.

[11] Brooks F., *Mythical man-month, or How software systems are created*. SPb.: Symbol-Plus, 2021.

[12] G. Sellitto, E. Iannone, Z. Codabux, V. Lenarduzzi, A. De Lucia, F. Palomba, and F. Ferrucci, "Toward understanding the impact of refactoring on program comprehension," in *29th International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2022, pp. 1–12.

[13] M. Di Penta, G. Bavota, and F. Zampetti, "On the relationship between refactoring actions and bugs: a differentiated replication," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 556–567.

[14] R. Turner, "Programming languages as technical artifacts," *Philosophy & technology*, vol. 27, no. 3, pp. 377–397, 2014.

[15] R. d. Lemos, D. Garlan, C. Ghezzi, H. Giese, J. Andersson, M. Litoiu, B. Schmerl, D. Weyns, L. Baresi, N. Bencomo *et al.*, "Software engineering for self-adaptive systems: Research challenges in the provision of assurances," *Software Engineering for Self-Adaptive Systems III. Assurances*, pp. 3–30, 2017.

[16] R. Turner, "Computational artifacts," in *Computational artifacts*. Springer, 2018, pp. 25–29.

[17] Golenkov, V. V., "Methodological problems of the current state of works in the field of artificial intelligence," *Open Semantic Technologies for Intelligent Systems = Open Semantic Technologies for Intelligent Systems (OSTIS-2021): collection of scientific papers / Belarusian State University of Informatics and Radioelectronics*, pp. 17–24, 2021.

[18] R. Turner and A. H. Eden, *Towards a programming language ontology*. na, 2007.

[19] C. Olteanu, "Programming, mathematical reasoning and sense-making," *International Journal of Mathematical Education in Science and Technology*, vol. 53, no. 8, pp. 2046–2064, 2022.

[20] F. W. Neiva, J. M. N. David, R. Braga, and F. Campos, "Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature," *Information and Software Technology*, vol. 72, pp. 137–150, 2016.

[21] O. Chaparro, G. Bavota, A. Marcus, and M. Di Penta, "On the impact of refactoring operations on code quality metrics," in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 456–460.

[22] N. N. Skeeter, N. V. Ketko, A. B. Simonov, A. G. Gagarin, and I. A. Tislenkova, "Artificial intelligence: Problems and prospects of development," in *13th International Scientific and Practical Conference-Artificial Intelligence Anthropogenic nature Vs. Social Origin*. Springer, 2020, pp. 306–318.

[23] Golenkov V.V., Gulyakina N.A., Davydenko I.T., Shunkevich D. V., Eremeev A.P., "Ontological design of hybrid semantically compatible intelligent systems based on the semantic representation of knowledge," in *Ontologiya proyektirovaniya*, Golenkov V.V., Ed. Russian Federation, Samara: Samara National Research University named after Academician S.P. Korolev, 2019, pp. 132–148.

[24] T. S. Dillon, E. Chang, and P. Wongthongtham, "Ontology-based software engineering-software engineering 2.0," in *19th Australian Conference on Software Engineering (ASWEC 2008)*. IEEE, 2008, pp. 13–23.

[25] D. C. Sales, L. B. Becker, and C. Koliver, “The systems architecture ontology (sao): an ontology-based design method for cyber–physical systems,” *Applied Computing and Informatics*, 2022.

[26] S. Elnagar, V. Yoon, and M. A. Thomas, “An automatic ontology generation framework with an organizational perspective,” *arXiv preprint arXiv:2201.05910*, 2022.

[27] A. H. Eden and R. Turner, “Problems in the ontology of computer programs,” *Applied Ontology*, vol. 2, no. 1, pp. 13–36, 2007.

[28] P. Lando, A. Lapujade, G. Kassel, and F. Fürst, “Towards a general ontology of computer programs,” in *International Conference on Software and Data Technologies*, vol. 2. SCITEPRESS, 2007, pp. 163–170.

[29] ——, “An ontological investigation in the field of computer programs,” in *Software and Data Technologies*. Springer, 2007, pp. 371–383.

[30] M. J. Jacobs, “A software development project ontology,” Master’s thesis, University of Twente, 2022.

[31] E. Tin, V. Akman, and M. Ersan, “Towards situation-oriented programming languages,” *ACM Sigplan Notices*, vol. 30, no. 1, pp. 27–36, 1995.

[32] H. Schiitze, “The prosit language v0. 4,” *Manuscript, Center for the Study of Language and Information, Stanford University, Stanford, CA*, 1991.

[33] A. Black, “An approach to computational situation semantics,” Ph.D. dissertation, PhD thesis, Department of Artificial Intelligence, University of Edinburgh . . . , 1993.

[34] W. J. Rapaport, “Syntax, semantics, and computer programs,” *Philosophy & Technology*, vol. 33, no. 2, pp. 309–321, 2020.

[35] J. Grimmelmann, “Programming languages and law: A research agenda,” *arXiv preprint arXiv:2206.14879*, 2022.

[36] Tetlow, Philip and Garg, Dinesh and Chase, Leigh and Mattingley-Scott, Mark and Bronn, Nicholas and Naidoo, Kugendran and Reinert, Emil, “Towards a semantic information theory (introducing quantum corollas),” 2022.

[37] Dijkstra E., *Programming Discipline*. M.: Mir, 1978.

[38] Reinhard Diestel, *Graph Theory*. Hamburg, Germany: Universität Hamburg, 2017.

[39] Kuznecov, O. P., *Diskretnaya matematika dlya inzhenera: Uchebnik dlya vuzov [Discrete Mathematics for an Engineer: A Textbook for High Schools]*. Moscow: Lan’, 2009.

[40] Golenkov, V. V., Gulyakina, N. A., Shunkevich, D. V., *Open technology for ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, Golenkov V.V., Ed. Minsk: Bestprint, 2021.

[41] X. Zhong, E. Cambria, and A. Hussain, “Does semantics aid syntax? an empirical study on named entity recognition and classification,” *Neural Computing and Applications*, vol. 34, no. 11, pp. 8373–8384, 2022.

[42] T.-D. Bradley, J. Terilla, and Y. Vlassopoulos, “An enriched category theory of language: from syntax to semantics,” *La Matematica*, pp. 1–30, 2022.

[43] F. Zhou, Y. Li, X. Zhang, Q. Wu, X. Lei, and R. Q. Hu, “Cognitive semantic communication systems driven by knowledge graph,” *arXiv preprint arXiv:2202.11958*, 2022.

[44] Kasyanov, V. N., Evstigneev, V. A., “Graphs in programming: processing, visualization and application,” *BHV–St. Petersburg*, p. 1104, 2003.

[45] Petrov, C. V., “Graphic grammars and automata (overview),” *Automation and telemechanics*, pp. 116–136, 1978.

[46] N. Sales and J. Efson, “An explainable semantic parser for end-user development,” Ph.D. dissertation, Universität Passau, 2022.

[47] Ford, Brian and Schiano-Phan, Rosa and Vallejo, Juan, *Component Design*, 11 2019, pp. 160–174.

[48] V. Kabilan, “Ontology for information systems (o4is) design methodology,” 2007.

[49] Y. I. Molorodov, “Development of information system based on ontological design patterns,” in *CEUR Workshop Proceedings*, 2019, pp. 26–30.

[50] Tuzov, V. A., “On the formalization of the task concept,” *M: Science*, pp. 73–83, 1986.

[51] Pospelov, D. A., “Situational management. theory and practice,” *M: Science*, p. 288, 2021.

[52] K. Lu, Q. Zhou, R. Li, Z. Zhao, X. Chen, J. Wu, and H. Zhang, “Rethinking modern communication from semantic coding to semantic communication,” *IEEE Wireless Communications*, 2022.

[53] Scott, M. L., *Programming Language Pragmatics*. Morgan Kaufmann publications, 2006.

[54] Scott, D., *Lattice Theory, Data Types and Formal Semantics, Formal Semantics of Programming Languages*. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[55] R. Lil, H. Zhu, and R. Banach, “Denotational and algebraic semantics for cyber-physical systems,” in *2022 26th International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 2022, pp. 123–132.

[56] Orlov, S.A., “Theory and practice of programming languages,” *St. Petersburg: Peter*, 2013.

[57] Ben-Ari M., *Programming languages. Practical Benchmarking*. M.: Mir, 2000.

[58] Gulyakina N.A., Pivovarchik O.V., Lazurkin D.A., “Languages and programming technology focused on the processing of semantic networks,” in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*. BSUIR, Minsk, 2012, pp. 222–228.

[59] D. Shunkevich, D. Koronchik, “Ontological approach to the development of a software model of a semantic computer based on the traditional computer architecture,” in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*. BSUIR, Minsk, 2021, pp. 75–92.

[60] Donald Knuth, *The art of programming. Volume 1. Basic Algorithms*, 2019.

Семантическая теория программ в интеллектуальных компьютерных системах нового поколения

Зотов Н.В.

Несмотря на активное развитие и использование языков программирования, общей теории программ, на основе которой можно было бы проектировать и разрабатывать прикладные системы, на данный момент не существует. В данной работе предлагается единая онтология языков программирования и представления программ на разных языках программирования. Работа показывает особенности представления программ и ключевые моменты процесса их интерпретации.

Received 28.10.2022

Non-procedural problem-solving models in next-generation intelligent computer systems

Maksim Orlov, Anastasia Vasilevskaya
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: orlovmassimo@gmail.com, vnastyap@gmail.com

***Abstract*—In the article, an approach to the design of problem solvers of intelligent systems based on non-procedural models is considered. The developed approach makes it possible to integrate any problem-solving models, including the principles of logical inference, to solve problems based on a general formal model.**

***Keywords*—Knowledge-driven systems; logical problem-solving models; logical graph languages; production problem-solving models; functional problem-solving models.**

I. Introduction

Currently, the usage of intelligent systems in a variety of fields is becoming increasingly relevant. Modern Artificial Intelligence technology is a whole family of various private technologies focused on the developing and maintening various types of components of intelligent computer systems that implement a variety of models for information representation and processing, different problem-solving models focused on the development of various classes of intelligent computer systems [1].

Modern intelligent computer systems consist of a knowledge base, a problem solver, and an intelligent interface. As the analysis of such systems shows, problem solvers do not have the proper level of semantic compatibility, are not able to fully coordinate their actions when solving complex problems and, in principle, solve problems in conditions when the problems are insufficiently formalized and the algorithm for solving them is unknown in advance [2]. In this article, an approach to the implementation of non-procedural problem-solving models in next-generation intelligent computer systems, based on ensuring compatibility between different models, is considered. The main attention is paid to logical problem-solving models, as an example of a procedural model, by analogy of which an approach to the design of any other models is implemented.

Logic solves the problems of proving the truth of propositions, argumentation of a proposition, the problem of generating and refuting hypotheses. When solving problems using logical models, it is possible to clearly trace the process of "reasoning" of the system, to obtain a protocol for solving the problem, which is an important knowledge. Obtaining new logic formulas based on existing ones is carried out by logical inference.

Frequently, in modern logical inference systems, components such as the rule base, working memory, and the logical inference mechanism are distinguished. These components have a strict boundary and are sometimes implemented in different programming languages and using different models of knowledge representation. This approach significantly limits the compatibility level for the subsystems of these computer systems and the level of compatibility of computer systems with each other as a whole.

Another problem of the current state of systems in which logical inference is implemented is that the semantics of the processed information is not taken into account. The system receives a certain set of logical rules, inference rules, and factographic statements as input and applies these rules on a working memory model (on a set of facts). Considering the semantics of the processed information allows not only to increase efficiency in solving problems using logical models but also to increase the level of negotiability and compatibility of computer systems.

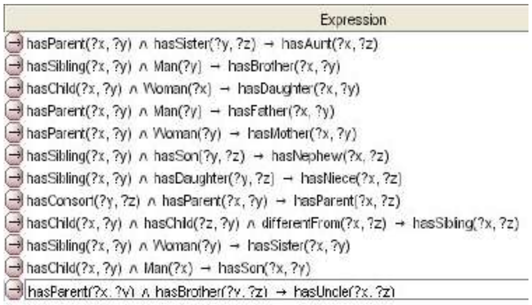
Likewise, no problem solvers have been developed that are able to combine different models for solving complex problems and ensure compatibility between them. Compatibility must be ensured not only within the same model, for example, compatibility of different logics, but also between different problem-solving models. When developing such problem solvers, it is important to notice not only the differences between different approaches, different logical models, but also their similarities.

The purpose of this work is not to develop a new problem-solving method or a new logic class, as well as to negate existing achievements in this field. The purpose of the work is to develop a model that allows integrating any problem-solving models and principles of logical inference for solving problems in intelligent systems based on a general formal model. In order to use any new or existing model, it is necessary to bring it to the formalism proposed in this article, which will allow integrating and synchronizing it with compatible components already available in the corresponding library.

II. Analysis of existing approaches to solving the problem

At the moment, many logical inference systems have been implemented [3], using the well-known rules of direct conclusion and resolution in various logic types, however, the problems of compatibility of the systems described above and collective problem solving using various problem-solving models remain relevant.

Each problem-solving model is defined by a language that provides a representation of a certain class of problem-solving methods in the memory of a cybernetic system and by an interpreter of these methods that defines the operational semantics of the specified language. It is necessary to consider the languages that can be used to set a logical problem-solving model. Such languages are Rule Interchange Format (RIF), Semantic Web Rule Language (SWRL), SHACL Rules, and Notation3 Rules, which are used in Semantic Web [4], [5]. In Figure 1, an example of rules in the SWRL language is represented.



Expression
hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
hasChild(?x, ?y) ∧ Woman(?x) → hasDaughter(?x, ?y)
hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x, ?z)
hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
hasChild(?x, ?y) ∧ hasChild(?z, ?y) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)
hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
hasChild(?x, ?y) ∧ Man(?x) → hasSon(?x, ?y)
hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)

Figure 1. Writing rules in the SWRL language

The described languages do not provide for the possibility of representing formulas in various logic types, so it is impossible to solve the described problems with them. Rule languages are specially built to infer conclusions. The syntax and semantics of ontology languages and rule languages are quite different, so the question arises how to combine them. There are several approaches, such as homogeneous and hybrid ones.

In a homogeneous approach, ontologies and rules are used on the same rights, i.e. a common language is created in which the same predicates are used both to express ontological statements and formulate rules (in particular, rules can be used to define classes and features of ontology). In this case, the problem of compatibility actually disappears, since the syntax and the interpretations become common – they only need to be extended to the rules, which is performed in a fairly standard way. The disadvantage of this approach is that combining different means in one language complicates its implementation greatly, and a homogeneous approach is often inapplicable, since ontologies and rule systems can be built independently by different specialists.

In a hybrid approach, the usual predicates, which are defined by rules (they can participate both in the conditions of rules and in their conclusions), and the predicates of ontologies, which are used as constraints in the conditions of rules, are strictly distinguished. The inference occurs through the interaction of individually implemented (existing) inference programs for rules and ontologies. The hybrid approach separates the builders of ontologies and rule systems from each other but also requires additional constraints to guarantee the solvability of the main problems for combinations of ontologies and rule systems (with solvable problems).

Semantic networks are convenient for representing knowledge of any kind, including logical formulas. The usage of semantic networks for deductive inference was researched by Quillian in 1966 [6]. He formally represented the semantics of natural language words and gave several examples of the inference technique. The deductive capabilities of Quillian were actually determined by the concept of “subclass” and the “modification” relation. The concept can be defined in terms of a more general concept and with the help of a modifying property, which is an “attribute – attribute value” combination.

An important technique used in semantic networks is a hierarchy, or classification system. In accordance with this technique, objects related to the subject domain are classified into a number of categories or classes based on their common properties. Using a hierarchical system in an extensive knowledge base of an intelligent system, it is especially convenient to use logical inference, since the inference that is valid for general concepts will be valid for particular concepts in relation to this general one. Despite the local success of such work, the systems remained static, non-extensible, and unable to be compatible.

Another important technique used in logical inference on semantic networks is knowledge localization [7]. The essence of localization is the possibility to identify an area of the semantic network in which subject knowledge are located (for example, constants, instances of classes), suitable for usage in logical inference premises. Taking into account the hierarchy of the knowledge base, it becomes most convenient to allocate a universe of reasoning, exceeding the scope of which is not advisable. Thus, the range of values of variables contained in the premises of logical formulas is limited, which allows reducing significantly the cost of searching in large knowledge bases.

Fuzzy inference systems [8], [9] are quite popular at the moment, whose semantic compatibility was also not considered.

III. Proposed approach

As part of this work, it is proposed to use an *OSTIS Technology* [10] as a basis, the principles of which make it possible to implement not just logical, production, functional, and other problem-solving models but also to ensure their compatibility, to implement a problem solver capable of combining various problem-solving models, including various logic types, to lay the foundation for creating interoperable computer systems.

The systems developed on the basis of the OSTIS Technology are called ostis-systems. The OSTIS Technology is based on a universal way of semantic representation of information in the memory of intelligent computer systems, called an *SC-code*. SC-code texts are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, depending on orientation, can be *sc-arcs* or *sc-edges*). The *Alphabet of the SC-code* consists of five main elements, on the basis of which SC-code constructions of any complexity are built, including more specific types of sc-elements (for example, new concepts). The memory that stores SC-code constructions is called semantic memory, or *sc-memory*.

The main advantage of using the SC-code for formalization and processing of logical formulas is that it provides compatibility between different problem-solving models. Any ostis-system has a problem solver, and there are problems for which the algorithm for solving them is unknown in advance and for which there is no ready-made method. The system must think and determine which agents can be involved in solving a particular problem.

The SC-code allows describing the relations between concepts of any form and complexity, which makes it a suitable option for using logical inference in next-generation intelligent computer systems, as well as using the hierarchy technique due to the ontological approach underlying the ostis-systems knowledge bases.

Within the technology, several universal variants of visualization of *SC-code* constructions are proposed, such as *SCg-code* (graphic variant), *SCn-code* (nonlinear hypertext variant), *SCs-code* (linear string variant).

Within this article, fragments of structured texts in the SCn and SCg codes [11] will often be used, which are simultaneously fragments of the source texts of the knowledge base, understandable to both human and machine. This allows making the text more structured and formalized, while maintaining its readability.

The basis of the knowledge base within the OSTIS Technology is a hierarchical system of subject domains and ontologies. Based on this, in order to solve the above problems, it is proposed to implement the following hierarchy of integrated subject domains:

Subject domain of logical formulas, propositions, and formal theories
⇒ *private subject domain**:
- *Subject domain of logical languages*
- *Subject domain of logical inference*

Subject domain of logical languages
⇒ *private subject domain**:
Subject domain of the propositional logic language

Subject domain of the propositional logic language
⇒ *private subject domain**:
Subject domain of the predicate logic language

Subject domain of logical problem-solving models
⇐ *private subject domain**:
- *Subject domain of logical languages*
- *Subject domain of logical inference*

Inheritance of subject domains allows using the described logics and their components in the description of any logics. The basic concepts allow developers of an intelligent system to add new logics. To implement a specific logical problem-solving model, it is necessary to create a subject domain that will be private in relation to the *Subject domain of logical problem-solving models* and the subject domain of some *logical language*, for example, the propositional logic language, the predicate logic language, the language of fuzzy logic, and others.

The *Subject domain of logical formulas, propositions, and formal theories* defines the denotational semantics of logical formulas, propositions, and formal theories and contains a formal specification of concepts necessary for the formation of logical formulas and propositions of any logics, including traditional, fuzzy, plausible, temporal, default logics, and any others. Logical formulas and propositions are interpreted using the concepts described in the *Subject domain of logical problem-solving models*, which includes a model and implementation of abstract agents necessary for solving logical problems. This subject domain includes the specification of concepts such as logical inference, inference rules, equivalent transformations, and axiom schemes.

Next, we will consider in more detail the fragments of sc-models of these subject domains and ontologies.

IV. Logical graph SCL language

Modern logic studies formal languages that serve to express logical reasoning. A logical language is a formal language intended to reproduce logical forms of natural language contexts, as well as to express logical laws and ways of correct reasoning in logical theories constructed in a given language. Logic does not study how knowledge was obtained – it allows representing knowledge, as well as deducing new knowledge from existing one (that is, deducing new formulas of the same logic from existing logic formulas), and establishing the accuracy of reasoning.

The **SCL Language** is a sublanguage of the SC-code for writing logical statements [12]. The SCL Language is a graph-type logical language used by ostis-systems. The texts of the SCL language are homogeneous semantic networks that are texts of the SC language. The alphabet of the SCL language is not allocated separately, since the alphabet of the SC-code is used, in which any statements, phenomena, regularities, programs, and any other knowledge can be described. The SCL language allows writing the texts of the propositional logic language, predicate logic language, and any other logical languages. The SC-code is a metalanguage for both the SCL language and for itself, that is, it allows describing the meaning of formulas written in SCL. Many formal languages, unlike SC, are not extensive enough to be a metalanguage for themselves. The specificity of the SCL language allocation is that the texts of this language can be processed in a special way. Logical inference inference can be made over propositions of the SL language.

One of the important features of SCL is its ability to represent predicate logic language texts taking into account the semantics of these texts (propositions). The SCL language is naturally oriented to work in the formal system of the predicate logic language. The SC language allows writing any relations and correspondences in a graph representation. The predicate value from a certain set of sc-variables corresponds to the result of a search operation on the template of some sc-construction (found or not found), which includes sc-constants and/or sc-variables with the corresponding configuration of relations between them. An approach based on the SCL language for the representation of formulas provides an opportunity to write generality and existence quantifiers not explicitly (this is not prohibited but is superfluous). The existence quantifier is an "embedded" concept in the sense that if some sc-element is included in some sc-structure, then the corresponding concept exists in this sc-structure. Thus, the existence quantifier is imposed automatically (unless another quantifier is explicitly imposed) on those sc-variables that are included in atomic logical formulas. The generality quantifier is imposed by default (unless another quantifier is explicitly imposed) on variables included in the equivalence and implication connectives in accordance with the denotational semantics of logical languages.

Such features simplify logical inference in the predicate logic in the SCL language, since this eliminates the need to bring the proposition into the Skolem normal form due to built-in quantifiers and the need for unification procedures conditioned by the search operation of the sc-construction by template, in which the necessary substitutions of variables occur.

V. Examples of formalizing statements in the SCL language

A proposition is understood as a certain structure (which includes sc-constants from some subject domain and/or sc-variables) or a logical connective that can be interpreted as true or false within any subject domain.

proposition
⇒ *subdividing**:
{• *atomic proposition*
• *non-atomic proposition*
}
⇒ *subdividing**:
{• *factographic proposition*
• *logical formula*
}

logical formula
⇒ *subdividing**:
{• *atomic logical formula*
• *non-atomic logical formula*
}

The truth of a proposition is set by indicating whether the sign of this proposition belongs to a formal theory corresponding to a given subject domain. The falsity of a proposition is set by specifying the belonging of the negation sign of this proposition to this formal theory.

In Figure 2, an example of a logical formula that is true within one formal theory and false within another is represented.

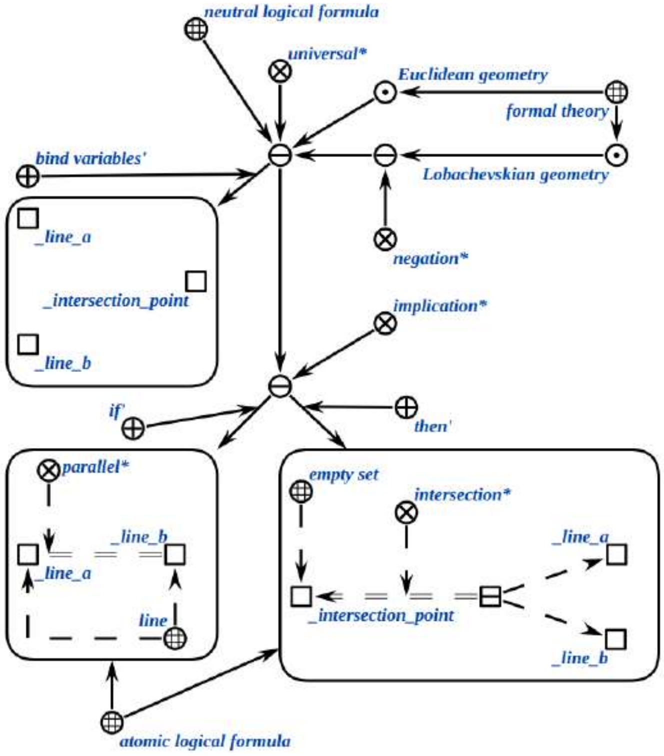


Figure 2. An example of a logical formula that is true within one formal theory and false within another

An *atomic logical formula* of the SCL language is interpreted as the set of all characters of some sc-text (sc-structure) containing at least one variable sc-element. Variables are free and bind subject variables that are intensional objects and are associated (have a value) with some constant element from the knowledge base. Figure 3 shows an example of an atomic logical formula that contains information about a triangle whose sine of the inner angle is equal to one.

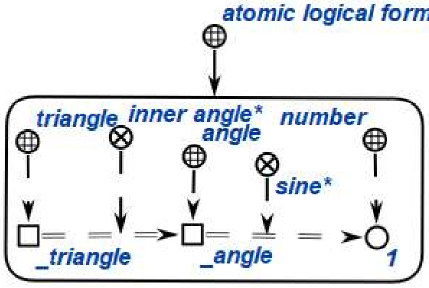


Figure 3. An example of formilizing an atomic logical formula

Each *non-atomic logical formula* of the SCL language is interpreted as a connective belonging to a relation corresponding to the type of non-atomic formula (conjunction, disjunction, negation, implication, equivalence, existence, universality) and linking the signs of the formulas included in the specified non-atomic formula. An example of a non-atomic logical formula is shown in Figure 4. This formula contains information that any triangle is either an acute triangle, or an obtuse triangle, or a right triangle.

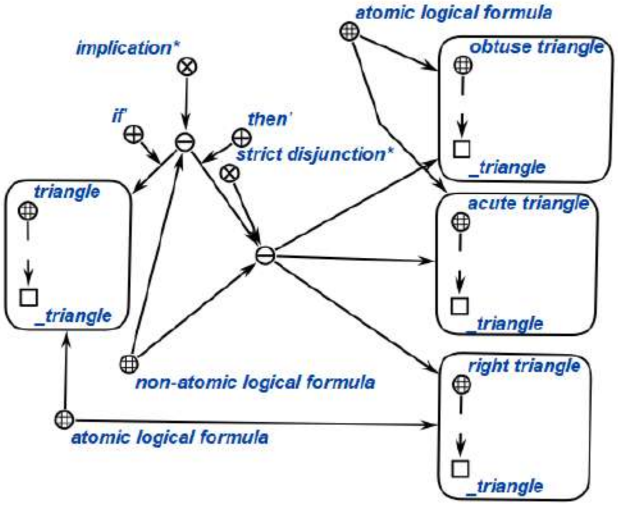


Figure 4. An example of formilizing a non-atomic logical formula

A statement is a semantic neighborhood of some logical formula, which includes the full text of this logical formula, as well as the fact that this logical formula belongs to some formal theory. The sign of a logical formula, the semantic neighborhood of which is a statement, is the main key sc-element within this statement. The signs of the concepts of the corresponding subject domain, which are part of any subformula of the specified logical formula, will be the key sc-elements within this statement.

The full text of some logical formula includes:

- the sign of this logical formula;
- signs of all its subformulas;
- elements of all logical formulas whose signs are included in this structure;
- all pairs of belonging that connect logical formulas whose signs are included in this structure with their components.

In Figure 5, there is an example of a statement that shows that the corresponding angles at the intersection of parallel lines of the secant are equal within the formal theory of Euclidean geometry.

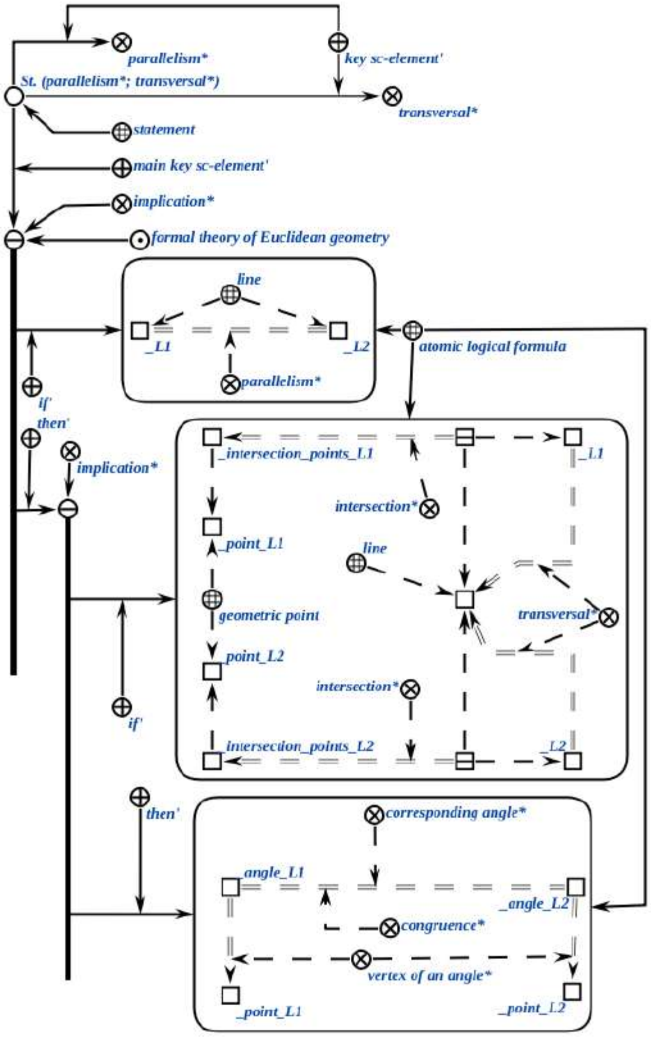


Figure 5. An example of the statement

A definition is a statement, the main key sc-element of which is a connective of equivalence that uniquely defines some concept based on other concepts. For the same concept within one formal theory, there may be several equivalence statements* that uniquely define some concept based on others, however, only one such statement within this formal theory can be marked as a definition. The remaining equivalence statements* can be interpreted as explanations of this concept.

In Figure 6, an example of a definition is given, which shows that a rhombus is a quadrilateral with all sides equal within the formal theory of Euclidean geometry.

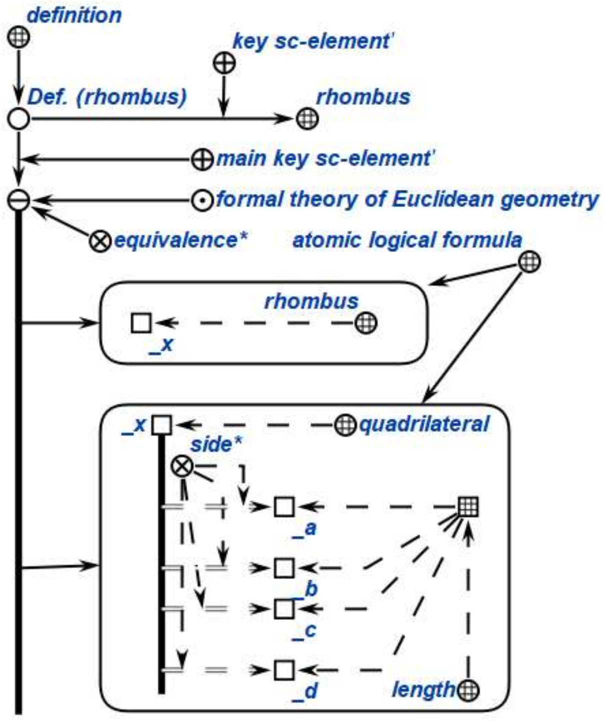


Figure 6. An example of a definition

VI. Machine of the SCL logical inference

An inference in a formal system is any sequence of formulas, so that any formula is either an axiom of this formal system, or a direct conclusion of any previous formulas according to one of the inference rules. The idea of deducibility is central to logic: in any formal axiomatic theory, a ‘theorem’ is a formula that is deduced from axioms. The correctness of conclusions is introduced and verified completely formally, without any connection with the truth of the premises included in it, i.e. exclusively from the point of view of the reasoning structure. From a practical point of view, the most important property of such formal correctness of reasoning is as follows: if we have managed to prove, using the methods of formal logic, the accuracy of the reasoning, and we know from experience that all the premises used are true, then we can be sure of the truth of the conclusion [13]. The truth of the premises used is set by the state of the knowledge base.

Various logical approaches allow designing problem solvers for intelligent systems in different subject domains, taking into account their specifics. The *Knowledge processing machine* for each specific system largely depends on the purpose of this system, the set of problems to be solved, the subject domain, and other factors. Some operations required in one subject domain will be redundant in another. For example, in a system that solves problems in geometry, chemistry, and other natural sciences, the usage of deductive inference methods will be reasonable, since the solution of problems in such subject domains is based only on reliable rules. In systems of medical diagnostics, for example, a situation constantly arises when a diagnosis can only be made with a certain degree of confidence and there can be no absolutely reliable answer to the question posed. In this regard, there is a need to use different knowledge processing machines in different systems, while the composition and capabilities of the knowledge processing machine in a particular system is determined not only directly by the developer but requires consultations with experts in this subject domain. Nevertheless, the basis for all logic types is classical logic, and its most general methods extend to other logics with some modifications, clarifications, and limitations.

Let us give a brief classification of existing logical problem-solving methods:

- **Classical deductive inference.** Classical deductive inference is the most popular in the building of automatic problem solvers, since it always gives a reliable result. Deductive inference includes direct, reverse, and logical inference (the resolution principle, the Erbran procedure, etc.) [13], all kinds of syllogisms [14], etc. The main problem of deductive inference is the impossibility of its usage in a number of cases when there is no reliable knowledge.
- **Inductive inference.** Inductive inference provides an opportunity to use various assumptions in the decision process, which makes it convenient for usage in poorly and difficultly formalizable subject domains, for example, in the building of medical diagnostic systems. The principles of inductive inference are discussed in detail in [15], [16].
- **Abductive inference.** In artificial intelligence, an abductive inference is usually understood as the inference of the best abductive explanation, i.e. the explanation of some event that has become unexpected for the system. Moreover, the “best” explanation is such one that satisfies special criteria determined depending on the problem being solved and the formalization used. The abductive inference is discussed in detail in [17], [18].
- **Fuzzy logic.** The theory of fuzzy sets and, accordingly, fuzzy logic is also used in systems related to difficultly formalizable subject domains [19], [20].

The theory of fuzzy logic is discussed in more detail in [9] and other publications.

- **Default logic.** The default logic is used, among other things, in order to optimize the reasoning process, supplementing the process of reliable inference with probabilistic assumptions in cases where the probability of error is extremely small. The default logic is discussed in more detail in the articles [21], [22].
- **Temporal logic.** The usage of temporal logic is very relevant for non-static subject domains in which the truth of a statement changes over time, which significantly affects the course of solving a problem [23], [24]. It should be noted that the knowledge representation language used in this work provides all the necessary capabilities for describing such dynamic subject domains.

A formal clarification of various information processing models in graphodynamic associative memory is **abstract graphodynamic associative machines**. The models of information processing, in particular, include models of parallel processing of knowledge corresponding to different logics and strategies for solving problems [25].

The advantage of using graphodynamic associative machines as a tool for creating next-generation intelligent computer systems is conditioned by the following aspects:

- the associative method of access to processed information is implemented in a fundamentally simpler way;
- it is much easier to maintain the open character of both the machines themselves and the formal models implemented on them;
- they are a convenient basis for the integration of various information processing models.

The other advantages of graphodynamic associative machines are conditioned by the advantages of graph texts and graph languages.

An **abstract scl-machine** is a logical inference machine, which belongs to the class of abstract sc-machines [12]. The internal language of the scl-machine is the above-mentioned SCL logical graph language, its operations correspond to the rules of logical inference. The family of specialized abstract graphodynamic knowledge processing machines is a formal clarification of the operational semantics of the above-mentioned specialized graph knowledge representation languages, each of which corresponds to one or more abstract machines.

These abstract machines correspond to different problem-solving models, different logics, different models of plausible reasoning. An agent from a family of logical inference agents can represent any inference rule that can be applied to solve a logical problem. In addition, agents are needed to perform equivalent transformations of a logical formula (for example, to write an equivalence formula as a conjunction of two disjunctions) and other agents that help apply inference rules on a set of logic language formulas.

Abstract scl-machine
⇒ *decomposition of an abstract sc-agent**:
{• *Abstract sc-agent for applying the inference rule*
• *Abstract sc-agent of equivalent transformations of a logical formula*
• *Abstract sc-agent of direct logical inference*
• *Abstract sc-agent of reverse logical inference*
}

The purpose of an Abstract sc-agent for applying the inference rule is to apply a given inference rule with given logical formulas. This sc-agent is activated when an initiated action belonging to the class *action of applying the inference rule* appears in the sc-memory. After the sc-agent checks the initiation condition, the process of applying the inference rule is performed, which consists in checking whether there are structures in the sc-memory that correspond to the condition for applying this rule and generating sc-constructions in accordance with the applied rule. The Agent of applying the inference rule is often used in the operation of direct inference and reverse inference agents, as well as others. An example of an inference rule can be the Modus ponens rule shown in Figure 7.

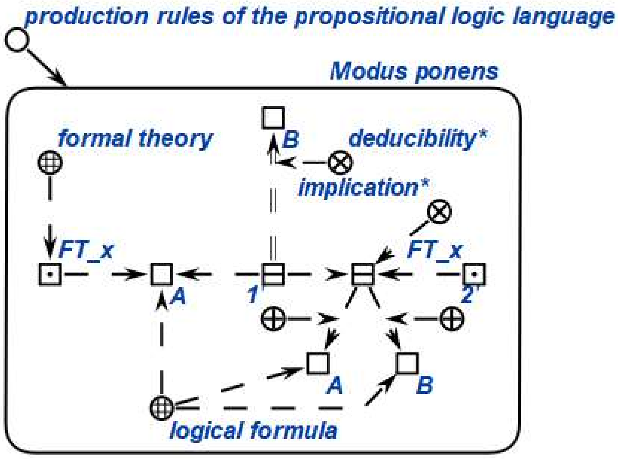


Figure 7. Formalization of the Modus ponens inference rule

The purpose of an Abstract sc-agent of equivalent transformations of a logical formula is to apply certain rules that bring the logical formula into a certain form. This sc-agent is activated when an initiated action belonging to the class *action of equivalent transformation of a logical formula* appears in the sc-memory. After the sc-agent checks the initiation condition, the process of converting the formula from one form to another is performed, while no new knowledge is generated in the sc-memory from the point of view of the subject domain under consideration. The response of this agent is a set of formulas that are equivalent in meaning but different in form of representation. As such forms, for example, conjunctive normal form or disjunctive normal form can serve. The Agent of equivalent transformation is often called during the operation of the agent for applying the inference rule, since logical formulas are not always in the form that is available for applying a particular inference rule but can be brought to the required form.

The purpose of an Abstract sc-agent of direct logical inference is to generate new knowledge based on some logical statements. This sc-agent is activated when an initiated action belonging to the class *direct logical inference action* appears in sc-memory. After the sc-agent checks the initiation condition, the process of direct logical inference is performed, which consists of cyclic operations of applying inference rules, generating new knowledge in sc-memory, and checking some condition, for example, the appearance of sc-elements from the target sc-structure in memory [26]. The input arguments of such an agent are the target structure, a set of formulas that are used during the inference by the agent of applying the inference rules, a set of inference rules, an input structure, and an output structure. As a result of performing the action by the agent of logical inference, an sc-structure is formed in the sc-memory, which is a decision tree. This tree consists of a sequence of nodes representing the applied rules that led to the appearance of the required knowledge in the sc-memory. Such a tree may be empty if the required structure could not be generated during logical inference. Figure 8 shows an example of the specification of the agent of direct logical inference.

The purpose of an Abstract sc-agent of reverse logical inference is to test hypotheses. Some hypotheses can be refuted, but by extracting the reasons why the hypothesis is refuted, it is possible to change the premise of the hypothesis so as to create a new hypothesis that can later become a useful theorem. This sc-agent is activated when an initiated action belonging to the class *reverse logical inference action* appears in sc-memory. After the sc-agent checks the initiation condition, the process of reverse logical inference is performed, which is similar to the process of direct logical inference, except that the search for rules is based not on the premises of formulas but on their conclusions [26]. The response of this agent will also be an inference tree showing which rules can be used to prove or refute the hypothesis put forward.

Abstract sc-agent of equivalent transformations of a logical formula

⇒ *decomposition of an abstract sc-agent**:

{

- *Abstract sc-agent of transforming a formula into a conjunctive normal form*
- *Abstract sc-agent of transforming a formula into a disjunctive normal form*
- *Abstract sc-agent for applying de Morgan's laws*
- *Abstract sc-agent of equivalent transformations of a logical formula by definition*
- *Abstract sc-agent of applying the negation properties of logical formulas*
- *Abstract sc-agent of applying the law of idempotence of logical formulas*
- *Abstract sc-agent of applying the law of commutativity of logical formulas*
- *Abstract sc-agent of applying the law of associativity of logical formulas*
- *Abstract sc-agent of applying the law of absorption of logical formulas*
- *Abstract sc-agent of applying the law of contradiction of logical formulas*
- *Abstract sc-agent of applying the law of double negation of logical formulas*
- *Abstract sc-agent of applying the law of splitting logical formulas*

}

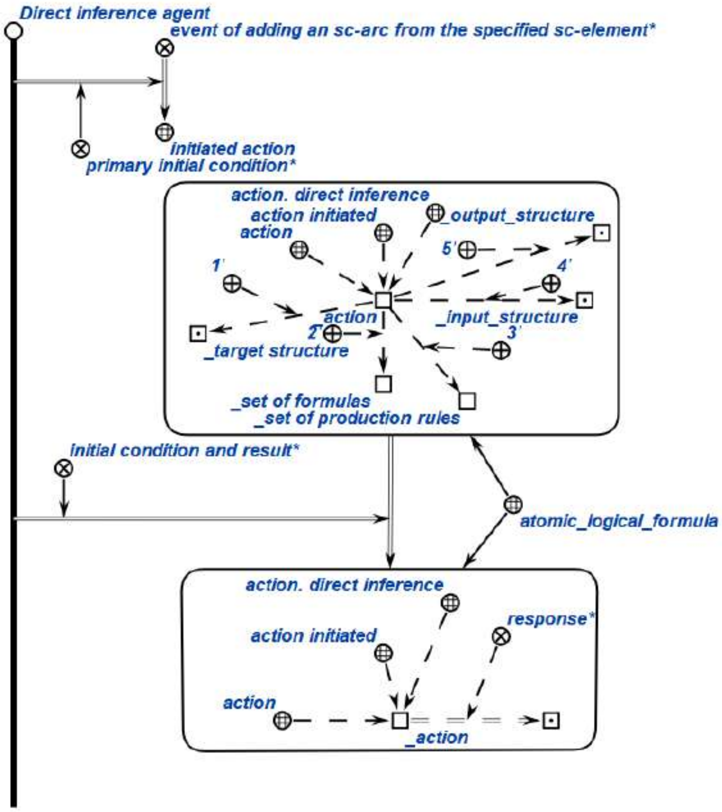


Figure 8. The specification of the agent of direct logical inference

VII. Example of formal inference in the SCL language

With the help of *resolution rules*, it is possible to effectively prove the formulas of the propositional logic language. Any formula is equivalent to some formula in conjunctive normal form, and therefore it is sometimes convenient to apply the resolution rule. Using equivalent transformations, it is also possible to obtain formulas suitable for using the resolution rule. Figure 9 shows the formalization of the resolution rule.

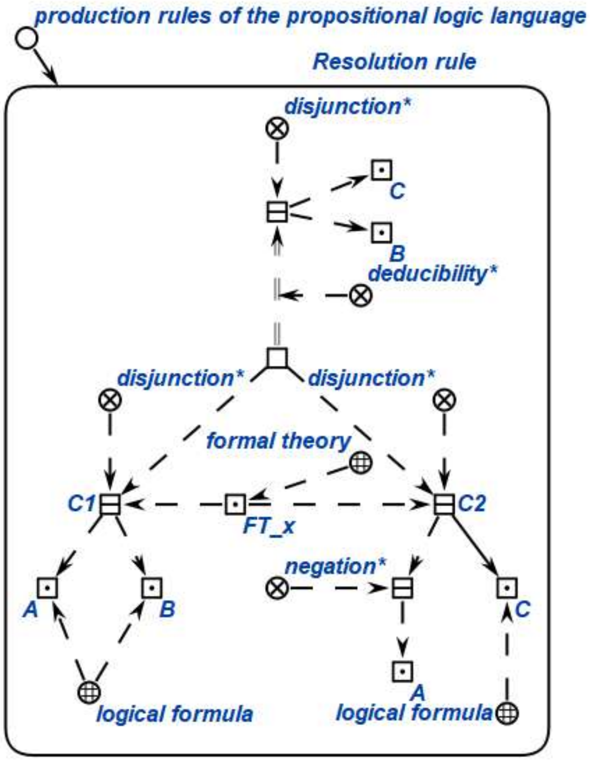


Figure 9. Formalization of the resolution rule

If any two disjuncts C_1 and C_2 have a pair of formulas A and $\neg A$, then a new disjunct can be formed from the remaining parts of the original disjuncts.

Let us give an example of the inference of a formula from a set of premises using the resolution principle [13].

If team A wins a football game, then city A' triumphs, and if team B wins, then city B' will triumph. Either only city A' or only city B' can win. However, if team A wins, then city B' does not triumph, and if team B wins, then city A' does not triumph. Consequently, city B' triumphs if and only if city A' does not triumph. The goal is to make sure that city B' triumphs if and only if city A' does not triumph.

Proving the inference of a formula is equivalent to proving the inconsistency of the inference of the negation for this formula. When using the resolution rule, this is especially convenient to use.

The formalization of logical formulas corresponding to the example is shown in Figure 10. Each non-atomic formula in the figure belongs to some formal theory, that is, is considered true.

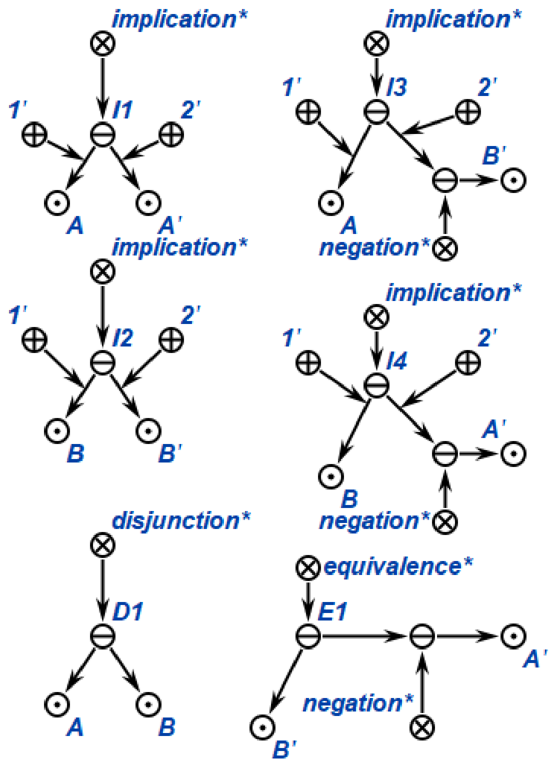


Figure 10. Formalization of rules for applying the resolution rule

Structure A is an atomic logical formula that contains the information "team A won", structure A' represents the formula denoting the triumph of city A'. Accordingly, the same is true for structures B and B'. First of all, it is necessary to bring the implication into conjunctive normal form according to the formula shown in Figure 11 and the equivalence by definition.

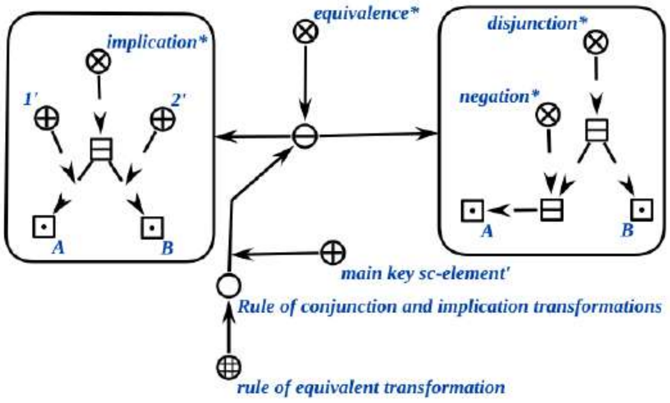

implication*
1'
2'
A
B
equivalence*
disjunction*
negation*
A
B
main key sc-element'
Rule of conjunction and implication transformations
rule of equivalent transformation

Figure 11. Formalization of rules for applying the resolution rule

Let us also apply negation to the formula that needs to be derived (equivalence). As a result, we obtain the following formulas (Fig. 12).

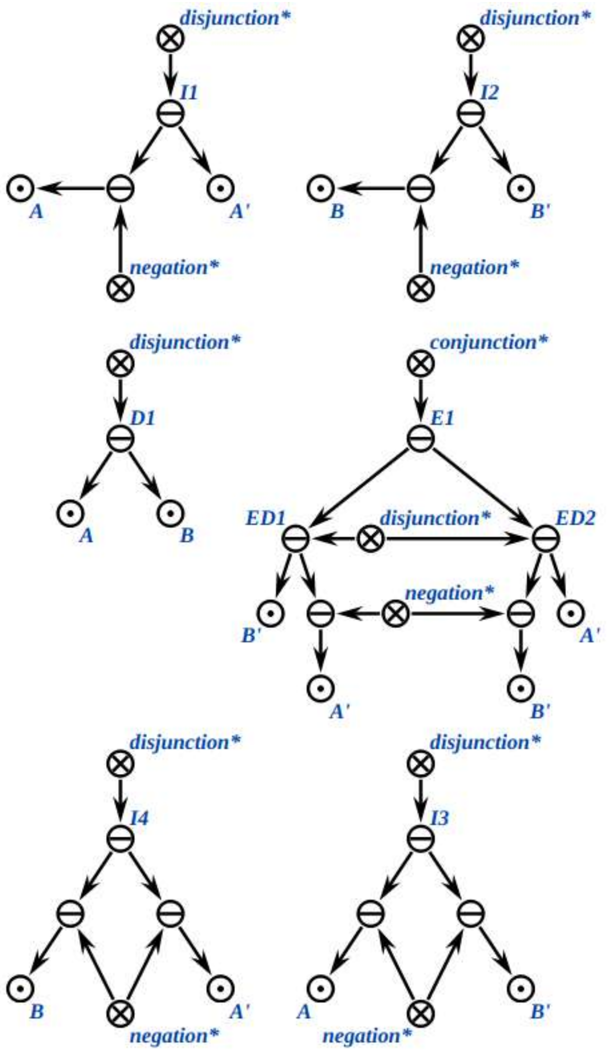


Figure 12. Formalization of rules for applying the resolution rule after conversion to conjunctive normal form

Further, applying the resolution rule for transformed formulas, we obtain an empty disjunction, which indicates the inconsistency of the set of formulas and proves the equivalence formula that city B' triumphs if and only if city A' does not triumph (Fig. 13 and 14).

VIII. Integration of Production and Functional Problem-Solving Models

Frequently, all the knowledge that a human operates with and that can be stored in the memory of an intelligent system can be divided into declarative and procedural. Declarative knowledge contains information about some objects, their features, properties, characteristics, the inclusion of objects among themselves in certain relationships, situations in which objects participate, the phenomena

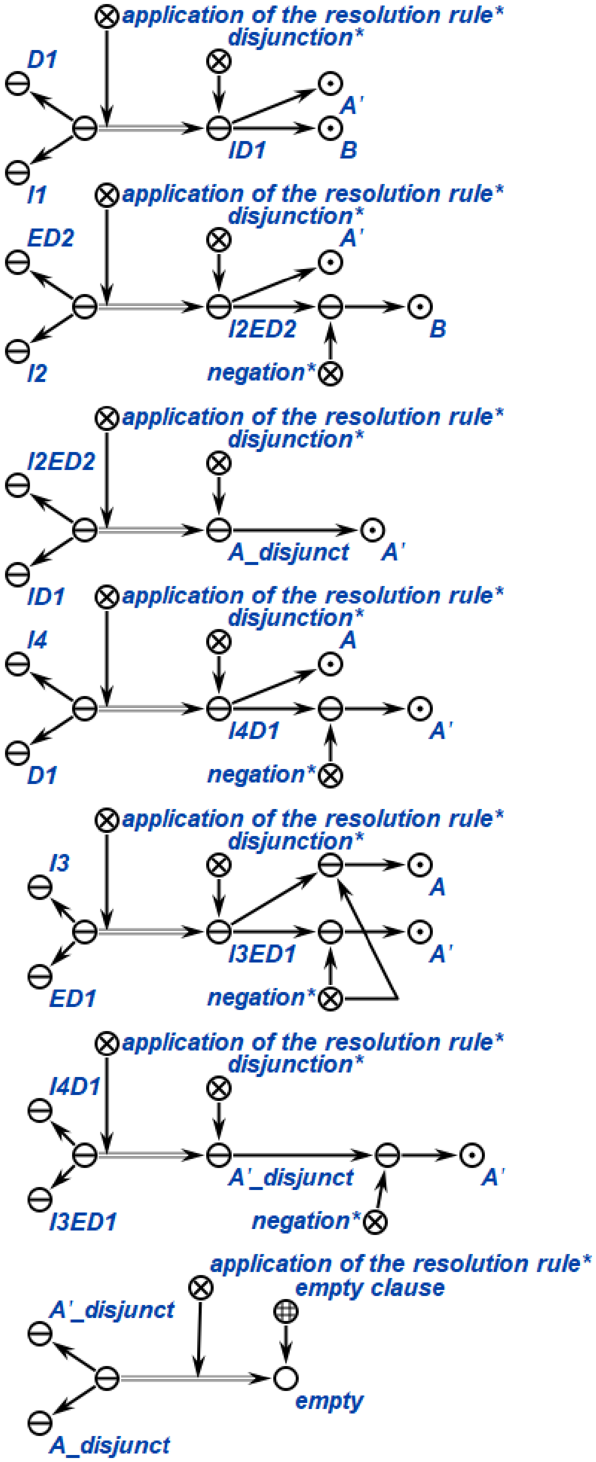

application of the resolution rule*
disjunction*
D1
I1
ID1
A'
B
application of the resolution rule*
disjunction*
ED2
I2
A'
I2ED2
B
negation*
application of the resolution rule*
disjunction*
I2ED2
ID1
A_disjunct
A'
application of the resolution rule*
disjunction*
I4
D1
A
I4D1
A'
negation*
application of the resolution rule*
disjunction*
I3
ED1
I3ED1
A
A'
negation*
application of the resolution rule*
disjunction*
I4D1
I3ED1
A'_disjunct
A'
negation*
application of the resolution rule*
empty clause
A'_disjunct
A_disjunct
empty

Figure 13. Application of the resolution rule

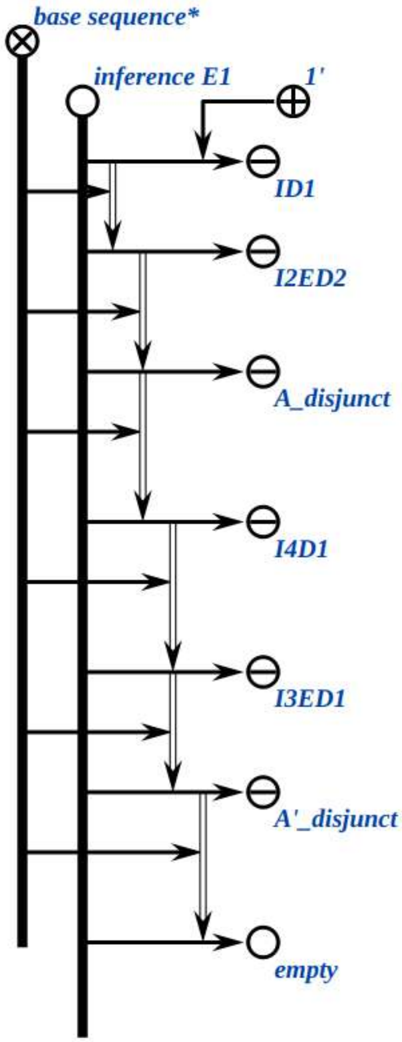


Figure 14. The result of applying the resolution principle

of reality, and its basic laws. Procedural knowledge allows the system to learn how to use certain declarative knowledge.

One of the ways to represent knowledge is a logical approach. Generalized knowledge about reality can be represented in the form of a formula of some calculus. However, even the simplest statements in natural language are not so easy to translate into the logic language, preserving the entire content of the text. Logical calculus is not suitable for displaying the totality of knowledge in intelligent systems.

Another way to describe knowledge is to use relational-type models. In such models, information units corresponding to objects, phenomena, facts, or processes are explicitly allocated.

The third way to describe knowledge is to use mixed-type models in which declarative and productive components are simultaneously present. Traditionally, this type of model includes frames and productions working on semantic networks. The production model is a development of the logical model. Production systems can be shown as transition graphs, which allows them to be represented in a natural way on the SC-code. Production systems often use an approach based on a "bulletin board", which is implemented within the principles of the OSTIS Technology.

Production systems have the following advantages [27]:

- productions describe a variety of knowledge in simple structures with a high degree of standardization;
- production systems satisfy the modularity principle to a high degree. Any production with software implementation can be considered as an independent module, the addition of which to the production system and its withdrawal from it occurs without additional costs;
- production systems simplify the organization of parallel processes in which all productions included in the scope of ready-made ones can be performed independently of each other.

Functional problem-solving models are based on the concept of a function as a fairly general mechanism for representing and analyzing problem solving. In this case, the calculation model is implemented without states. A functional program cannot change the data it already contains but can only generate new ones. A neural network problem-solving model is a particular case of a functional problem-solving model. The advantages of functional methods are:

- high reliability due to clear structuring of data and functions;
- great capabilities for parallel computing.

The representation of functional models is also unified using the OSTIS Technology, and such models can be integrated with any other models when solving complex problems.

IX. Conclusion

In the article, the implementation of non-procedural problem-solving models of intelligent systems based on the OSTIS Technology is proposed, which makes it possible to realize compatibility between different problem-solving models and allow intelligent systems to solve complex problems. The hierarchy of complex subject domains necessary to achieve the set goals is designed.

The operational semantics of logical languages has been clarified in the form of a specification of the corresponding abstract sc-agents.

An example of the formalization of logical formulas, as well as the process of logical inference using semantic networks, is given.

The results obtained will allow structuring existing logics and using various approaches of non-procedural models in solving complex problems.

References

[1] Akshita Rastogi, Shivam, Rekha Jain, “Risk and challenges in intelligent systems,” *Proceedings of the Third International Conference on Information Management and Machine Intelligence, ICIMMI 2021*, 2022.

[2] Martin Molina, “What is an intelligent system?” 2022.

[3] Peter Flach, Kacper Sokol, “Simply logical – intelligent reasoning by example (fully interactive online edition),” 2022.

[4] J. M. Giménez-García, A. Zimmermann, and P. Maret, “Ndfluents: An ontology for annotated statements with inference preservation,” 2017.

[5] Abdur Rakib, Abba Lawan, “The Semantic Web rule language expressiveness extensions – a survey,” *Ontology-driven CropBase knowledge system*, 2019.

[6] Apatova N., Gaponov A., Smirnova O., “The possibilities of Artificial Intelligence in teaching higher mathematics,” 2021.

[7] Vadim Moshkin, Nadejda Yarushkina, “Modified knowledge inference method based on fuzzy ontology and base of cases,” *Creativity in Intelligent Technologies and Data Science*, 2019.

[8] Stefania Tomasiello, Witold Pedrycz, Vincenzo Loia, “Fuzzy inference systems,” *Contemporary Fuzzy Logic, A Perspective of Fuzzy Logic with Scilab*, 2022.

[9] Uehara, Kiyohiko and Hirota, Kaoru, “Fuzzy inference: Its past and prospects,” *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 21, pp. 13–19, 01 2017.

[10] Golenkov Vladimir and Guliakina Natalia and Shunkevich Daniil, *Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[11] (2022, September) IMS.ostis Metasystem. [Online]. Available: https://ims.ostis.net

[12] Golenkov V., Korolev V., “Basic transformations of SQL language texts for the implementation of deductive inference mechanisms,” *Minsk: ITC of NAS of Belarus*, 1996.

[13] Averin A.I. and Vagin V.N., “Using parallelism in deductive inference,” *Journal of Computer and Systems Sciences International*, vol. 43, pp. 603–614, 07 2004.

[14] Satya Sundar Sethy. (2021) Mediate inference (syllogism).

[15] Norton John, “A demonstration of the incompleteness of calculi of inductive inference,” *The British Journal for the Philosophy of Science*, vol. 70, pp. 1119–1144, 12 2019.

[16] Yini Zhang and Yilin Wang. (2022) Missing-edge aware knowledge graph inductive inference through dual graph learning and traversing.

[17] Abdul Rahman, Safawi and Ibrahim, Zaharudin and Paiman, Jailani and Bakar, Amzari and Mohd Amin, Zahari, “The decision processes of abductive inference,” *Advanced Science Letters*, vol. 21, pp. 1754–1757, 06 2015.

[18] Gungov, Alexander, “The ampliative leap in diagnostics: The advantages of abductive inference in clinical reasoning,” *History of Medicine*, vol. 5, pp. 233–242, 01 2018.

[19] Geramian A. and Mehregan M.R. and Garousi Mokhtarzadeh and N. and Hemmati M. (2017) Fuzzy inference system application for failure analyzing in automobile industry.

[20] Son L.H. and Van Viet P. and Van Hai P. (2017) Picture inference system: a new fuzzy inference system on picture fuzzy set.

[21] Lupea, Mihaiela, “DARR–a theorem prover for constrained and rational default logics,” vol. 1, 01 2002.

[22] Weydert, Emil, “Defaults, logic and probability – a theoretical perspective,” *KI – Künstliche Intelligenz, v.4/01, 44–49 (2001)*, 11 2022.

[23] Chen, Gang and Wei, Peng and Liu, Mei, “Temporal logic inference for fault detection of switched systems with Gaussian process dynamics,” *IEEE Transactions on Automation Science and Engineering*, vol. PP, pp. 1–16, 05 2021.

[24] Rybakov, V., “Multi-agent temporal nontransitive linear logics and the admissibility problem,” *Algebra and Logic*, vol. 59, 05 2020.

[25] Golenkov V., Gulyakina N., “Grapho-dynamic association models and facilities of parallel information handling in artificial intelligence systems,” *BSUIR Proseedings*, 2003.

[26] Gavrilova T.A., Horoshevski V.F., *Knowledge bases of intelligent systems*, 2000.

[27] Kuznetsov V. E., *Computer representation of informal procedures*, 1989.

Непроцедурные модели решения задач в интеллектуальных компьютерных системах нового поколения

Орлов М.К., Василевская А.П.

В работе рассматривается подход к проектированию решателей задач интеллектуальных систем на основе непроцедурных моделей. Разрабатываемый подход позволяет интегрировать любые модели решения задач, в том числе принципы логического вывода, для решения задач на основе общей формальной модели.

Received 30.10.2022

Convergence and integration of artificial neural networks with knowledge bases in next-generation intelligent computer systems

Mikhail Kovalev
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
michail.kovalev7@gmail.com

Aliaksandr Kroshchanka
Brest State Technical University
Brest, Belarus
kroschenko@gmail.com

Vladimir Golovko
Brest State Technical University
Brest, Belarus
vladimir.golovko@gmail.com

***Abstract*—In the article, an approach to the integration and convergence of artificial neural networks with knowledge bases in next-generation intelligent computer systems through the representation and interpretation of artificial neural networks in a knowledge base is considered. The syntax, denotational, and operational semantics of the language for representing neural network methods in knowledge bases are described. The stages of building of neural network problem-solving methods with the help of intelligent framework for designing artificial neural networks are described.**

***Keywords*—problem-solving model, ontological approach, neuro-symbolic AI, artificial neural network**

I. Introduction

The term of a next-generation intelligent computer system implies that such systems, among others, have the following capabilities [1]:

- the ability to constantly improve the quality of problem solving;
- the ability to acquire skills for solving fundamentally new problems;
- the ability to explain their own decisions;
- the ability to find and eliminate errors in their own decisions (the ability to introspect).

Ensuring the above abilities is fundamentally possible in the concept proposed by the OSTIS project [1] due to the unification of the representation and ontological structuring of knowledge describing *problems*, subject domains within which problems are solved, and *problem-solving methods*.

Representation of various problems-solving methods in a common knowledge base ensures the semantic compatibility of these methods. When solving a problem using such methods, the system does not interact with them on the principle of "inputs–outputs". On the contrary, a common memory allows real-time transformation of input knowledge using any available methods, which provides the ability to introspect and explain the decisions of the system.

Promising and actively developing problem-solving methods are artificial neural networks (ANN), which is determined, on the one hand, by the evolution of the theory of ANN and, on the other hand, by the hardware capabilities of the machines that are used to train them.

The advantages of ANN include the ability to solve problems with unknown patterns, as well as the ability to solve problems without the need to develop problem-oriented approaches.

However, most neural network models work like a "black box" [2], which is one of the main disadvantages of this problem-solving method. Modern problems increasingly require explanation of their solution. A whole direction of Explainable AI has appeared, within which various attempts are made to explain the decisions of ANN [3], [4]. Approaches that propose the integration of neural networks with knowledge bases are being developed [5]–[7].

As a disadvantage of ANN we can also name the heuristic nature of the process of finding optimal architectures of models and the parameters for their training, as well as the high requirements for the scope of knowledge of neural network models researchers.

Based on the above abilities, the presence of which must be ensured in next-generation intelligent computer systems, the problem of developing an approach to the integration of ANN into the knowledge base of an intelligent system arises, both as a problem-solving method and as an object of automatic design of new methods. The solution of this problem will allow overcoming the above disadvantages of the neural network method.

The purpose of the research is to expand the range of problems solved by intelligent systems by developing a set of models, methods, and tools for representing, designing, and processing artificial neural networks in intelligent systems and integrating them with other problem-solving models.

II. Proposed approach

The basis of the proposed approach is the usage of the OSTIS technology and its basic principles [8]. Intelligent systems developed using the OSTIS technology are called ostis-systems. Any ostis-system consists of a knowledge base, a problem solver, and a user interface.

The problem solver performes the processing of fragments of the knowledge base. At the operational level, processing means adding, searching, editing, and deleting sc-nodes and sc-connectors of the knowledge base. On the semantic level, such an operation is an *action performed in the memory of an action subject*, where, in the general case, the subject is an ostis-system and the knowledge base is its memory. An *action* is defined as the influence of one entity (or some set of entities) to another entity (or some set of other entities) according to some purpose.

Actions are performed according to the set problems. A *problem* is a formal specification of some action, sufficient to perform this action by some subject. Depending on a particular class of problems, it is possible to describe both the internal state of the intelligent system itself and the required state of the external environment [9].

Similar problems are grouped into classes, for which generalized problem formulations are described. The following classes of problems for ANN are defined [10]:

- *The classification problem*. The problem of constructing a classifier, i.e. a mapping $\tilde{c} : X \to C$, where $X \in \mathbb{R}^m$ is the feature space of the input example, $C = C_1, C_2, ...C_k$ is a finite and usually small set of class labels.
- *The regression problem*. The problem of constructing an evaluation function by examples $(x_i, f(x_i))$, where $f(x)$ is an unknown function. The *evaluation function* is a mapping of the form $\tilde{f} : X \to \mathbb{R}$, where $X \in \mathbb{R}^m$ is the feature space of e.a.p.
- *The clustering problem*. The problem of constructing a function $a : X \to Y$ that matches any object $x \in X$ with a cluster number $y \in Y$ with a certain distance metric $\rho(x, x')$, where *X* is a set of objects, *Y* is a set of cluster numbers (names, labels), $x, x' \in X$.
- *The problem of decreasing the dimensionality of the feature space*. The problem of constructing a function $h : X \to Y$ that preserves the given relations between points of sets X and Y, where $X \subset \mathbb{R}^p$, $Y = h(X) \subset \mathbb{R}^q$, $q < p$.
- *The control problem*. The problem of constructing a model-regulator for the state of a complex dynamic object.
- *The filtering problem*. The problem of building a model that cleans the original signal containing some noise and reduces the influence of random errors in the signal.
- *The detection problem*. It is a special case of the classification and regression problems. The problem of constructing a model that performs the detection of objects of certain types in photo and video images.
- *The problem with associative memory*. The problem of constructing a model that allows reconstructing the original example based on previously saved examples.

For classes of problems, classes of methods for their solution are formulated. A *problem-solving method* is defined as a problem-solving program of the corresponding class, which can be either procedural or declarative. In turn, a *class of problem-solving methods* is defined as a set of all possible problem-solving methods having a common language for representing these methods. The method representation language allows describing the syntactic, denotational, and operational semantics of this method.

In this article, we propose to consider ANN as a class of problem-solving methods with its own representation language. According to the OSTIS technology, the specification of a class of problem-solving methods is reduced to the specification of the corresponding method representation language, i.e. to the description of its syntactic, denotational, and operational semantics.

To achieve semantic compatibility with other problem-solving methods of the OSTIS technology, it is proposed to describe neural network methods within semantic memory, accordingly, the syntax of the representation language of neural network problem-solving methods is the syntax of the SC-code used in the OSTIS technology for knowledge representation.

Thus, in order to add neural network problem-solving methods to the stack of the OSTIS technology and thus expand the range of problems solved by ostis-systems, it is necessary to describe the denotational and operational semantics of the representation language for the neural network problem-solving method.

The denotational semantics of neural network method representation language is described within the subject domain and its corresponding ontology of a neural network method. This model is described in detail in **Section III**.

The operational semantics of any problem-solving method representation language is the specification of a family of agents providing the interpretation of any method belonging to the corresponding method class. This family is an interpreter of the corresponding problem-solving method. Within the OSTIS technology, such an interpreter is called a *problem-solving model*. Since the OSTIS technology uses a multi-agent approach, the development of a neural network problem-solving model is reduced to the development of an agent-oriented model of ANN interpretation. This model is described in **Section IV**.

A *skill* is a method, the interpretation of which can be fully carried out by a given cybernetic system, in

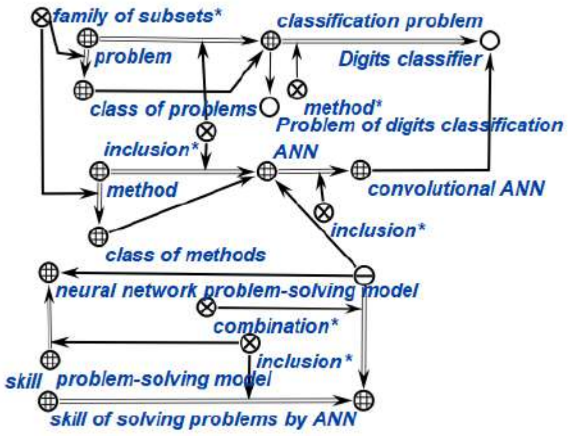


Figure 1. A fragment of the set-theoretic ontology of ANN

the memory of which the specified method is stored [9]. Thus, forming the specification for the neural network problem-solving method and neural network problem-solving model in the ostis-system, we can say that such system possesses the skill of problem solving with the help of ANN.

In Figure 1, a fragment of the ANN ontology is shown, describing the relation of such concepts and nodes as:

- a class of problems that can be solved by ANN (for example, the class of classification problems);
- a class of neural network problem-solving methods;
- a neural network problem-solving model;
- a skill in problem solving with the help of ANN;
- specific problems and methods of their solution (for example, a specific trained convolutional ANN).

The usage of ANN as a problem-solving method implies the usage of an already designed and trained ANN. However, the presence of a neural network method description language in ostis-system memory opens the way for automation of the design and training ANN processes themselves. Such automation is represented by separate classes of problems and the corresponding skills for their solution. The approach to such automation is described in **Section V**.

III. Denotational semantics of the neural network representation language

As it was already mentioned, the denotational semantics of neural network method representation language is described within the subject domain (SD) and its corresponding neural network method ontology. The SD of neural network methods is a private SD of the method.

The maximum class of artificial neural network research objects is an *artificial neural network*.

The SD of a neural network method and key elements of its ontology are described in [10]. In this article, an extension of the SD of neural network methods, described in [10], is represented.

Let us demonstrate an updated classification of neural network methods (the added classes are in bold):

artificial neural network

:= [neural network method]

⇐ _inclusion*_:
method

⇒ _subdividing*_:
Typology of ANN on the basis of the directivity of connections^

= {
- _ANN with direct connections_
  ⇒ _decomposition*_:
  {
  - _perceptron_
    ⇒ _decomposition*_:
    {
    - _Rosenblatt perceptron_
    - _autoencoder ANN_

    }
  - _support vector machine_
  - _ANN of radial basis functions_
  - ***convolutional ANN***

  }
- _ANN with inverse connections_
  ⇒ _decomposition*_:
  {
  - _Hopfield ANN_
  - _Hamming ANN_

  }
- _recurrent ANN_
  ⇒ _decomposition*_:
  {
  - _Jordan ANN_
  - _Elman ANN_
  - _multi-recurrent ANN_
  - _LSTM-element_
  - _GRU-element_

  }

}

⇒ _subdividing*_:
Typology of ANN on the basis of completeness of connections^

= {
- _fully connected ANN_
- _weakly connected ANN_

}

The concepts for describing metrics of neural network methods effectiveness are also added in the SD of neural network methods. These metrics are taken into account by the problem solver when deciding to use one or another neural network method.

Metrics can be classified according to the type of problem to be solved.

ANN quality assessment metric

⇒ *subdividing**:
*Metric typology by problems^*
= {● *classification metrics*
⇒ *decomposition**:
{● *ANN precision*
● *ANN completeness*
● *F1-metric*
}
● *regression metrics*
⇒ *decomposition**:
{● *MAE*
● *MAPE*
● *RMSE*
}
}

ANN precision

:= [precision]
:= [proportion of correctly identified positive outcomes in the total number of outcomes that were identified as positive]
⇒ *formula**:
[

$$PRE = \frac{TP}{TP + FP}$$

where *TP* and *FP* are the number of true-positive and false-positive predictions of the neural network, respectively]

ANN completeness

:= [recall]
:= [proportion of correctly identified positive outcomes in the total number of positive outcomes]
⇒ *formula**:
[

$$REC = \frac{TP}{TP + FN}$$

where *TP* and *FN* are the number of true-positive and false-negative predictions of the neural network, respectively]

F1-metric

⇒ *formula**:
[

$$F1 = 2 * \frac{PRE * REC}{PRE + REC}$$

where *PRE* and *REC* are the accuracy and completeness of ANN, respectively]

MAE

:= [mean absolute error]
⇒ *formula**:
[$\frac{1}{N}\sum_{i=1}^{N}|y^i_{etalon} - y^i_{predicted}|$, y^i_{etalon} – the reference value, $y^i_{predicted}$ – the value obtained by the ANN, *N* – the size of the training dataset]

MAPE

:= [mean absolute percentage error]
⇒ *formula**:
[$\frac{1}{N}\sum_{i=1}^{N}\frac{y^i_{etalon}-y^i_{predicted}|}{y^i_{etalon}} * 100\%$,
y^i_{etalon} – the reference value,
$y^i_{predicted}$ – the value obtained by the ANN,
N – the size of the training dataset]

RMSE

:= [root mean squared error]
⇒ *formula**:
[$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y^i_{etalon} - y^i_{predicted})^2}$, y^i_{etalon} – the reference value, $y^i_{predicted}$ – the value obtained by the ANN,
N – the size of the training dataset]

IV. Operational semantics of the neural network representation language

Operational semantics of neural network representation language is defined by the agent-oriented model of artificial neural network interpretation and specification of corresponding actions.

A neural network method is described in the form of a program in some programming language, which can be either external in relation to the ostis-system or internal (at the moment, an SCP language). Each such programming language corresponds to some private subject domain of the SD of neural network methods.

Subject domain of neural network methods

:= [Subject domain of artificial neural networks]
⇒ *private subject domain**:
{● *Subject domain of neural network methods in SCP*
● *Subject domain of neural network methods in Python*
● *Subject domain of neural network methods in C++*
}

In the case of description of a neural network method in an external language, such method is described in the corresponding subject domain, within which the action for interpretation of this method is also specified. This action corresponds to an agent implemented in the corresponding programming language.

However, to achieve convergence and integration, it is necessary to describe neural network methods in the internal language of the ostis-system, which is SCP [1].

An scp-program is a sequence of generalized specifications (templates) of scp-operators. Each scp-operator is an action in ostis-system memory (sc-memory). During interpreting an scp-program, the abstract sc-agent of creating scp-processes creates an scp-process, taking into account the specific scp-program interpretation parameters.

In many cases that means substituting arguments in the generalized scp-operator specifications of the program and generating specific instances of these programs (methods). Then, the initial operator is added to the set of real entities, and the program execution begins.

Thus, the interpretation of an scp-program comes down to agent-based processing of actions in the scp-memory, which are scp-operators.

The neural network method representation language in SCP is an extension of the SCP language. It is extended at the expense of actions, specific to the SD of ANN. The subject domain and its corresponding ontology of neural network methods in SCP describes the specification of actions for interpretation of ANN within ostis-system memory, which extend the range of standard scp-operators. The following hierarchy of such actions can be distinguished:

action for interpreting the ANN layer
⇒ *decomposition**:
{• *action for calculating the weighted sum of all neurons of the layer*
• *action for calculating the activation function for all neurons of the layer*
• *action for interpreting the convolutional layer*
• *action for interpreting the pooling layer*
}

To describe the specification of the above actions, it is necessary to introduce the concepts of *oriented number set* and *matrix* using which the input values of ANN, output values of ANN, weight matrices, and so on are specified.

Each element of an oriented number set is some number. The numbers can be represented as sc-nodes or with a string representation of the whole set, for which a special relation *string representation of the oriented number set** is used. This relation was introduced in order to optimize some implementation options of the agent interpreting action using the concept of oriented number set.

oriented number set
:= [oriented number set]
⇐ *inclusion**:
number
⇐ *inclusion**:
oriented set
⇐ *first domain**:
*string representation of an oriented number set**

A *matrix* is an oriented set of oriented sets of equal power numbers.

1. Action for calculating the weighted sum of all neurons of the layer

The (*objects'*) arguments of this action are set by the following relations:

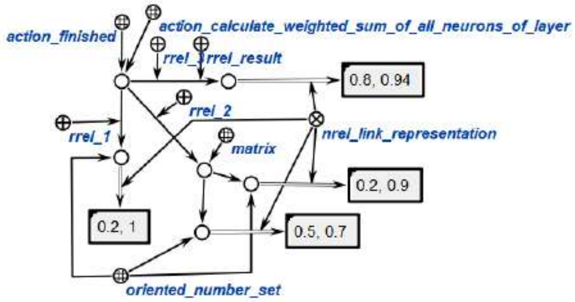


Figure 2. An example for the specification of the action for calculating the weighted sum of all neurons of the layer

input vector'
⇒ *first domain**:
action for interpreting the ANN layer
⇒ *second domain**:
oriented number set

matrix of neuron synapse weights of the layer'
⇒ *first domain**:
action for processing ANN
⇒ *second domain**:
matrix

The result of the (*result'*) action is an oriented number set, which is the weighted sum of neurons of the corresponding layer.

An example for the specification of the action for calculating the weighted sum of all neurons of the layer for a layer with two neurons and an input vector of dimension 2 is shown in Figure 2.

2. Action for calculating the activation function for all neurons of the layer

The arguments of this action are set by the following relations:

vector of weighted sums of layer neurons'
⇒ *first domain**:
action for processing ANN'
⇒ *second domain**:
oriented number set

threshold vector of layer neurons'
⇒ *first domain**:
action for processing ANN'
⇒ *second domain**:
oriented number set

activation function'
⇒ *first domain**:
action for processing ANN'
⇒ *second domain**:
function

The result of the action is an oriented number set, which are the output values of the layer neurons.

3. Action for interpreting the convolutional layer

The arguments of this action are set by the following relations:

input matrix'
⇒ *first domain**:
action for processing ANN
⇒ *second domain**:
matrix

convolution kernel'
⇒ *first domain**:
action for interpreting the convolutional layer
⇒ *second domain**:
matrix

convolution step'
⇒ *first domain**:
action for interpreting the convolutional layer
⇒ *second domain**:
number

The result of the action is the matrix resulting from the convolution of the input matrix with the convolution kernel.

4. Action for interpreting the pooling layer

The arguments of this action are defined by the following relations:

input matrix'
⇒ *first domain**:
action for processing ANN
⇒ *second domain**:
matrix

pooling window size'
⇒ *first domain**:
action for interpreting the pooling layer
⇒ *second domain**:
matrix

pooling window step'
⇒ *first domain**:
action for interpreting the pooling layer
⇒ *second domain**:
number

The result of the action is the matrix obtained as a result of pooling the input matrix.

If it is necessary to specify different arguments for neurons of the same layer, it is possible to specify the corresponding actions, however, this was not used in this work due to the poor knowledge of neural network models of this kind.

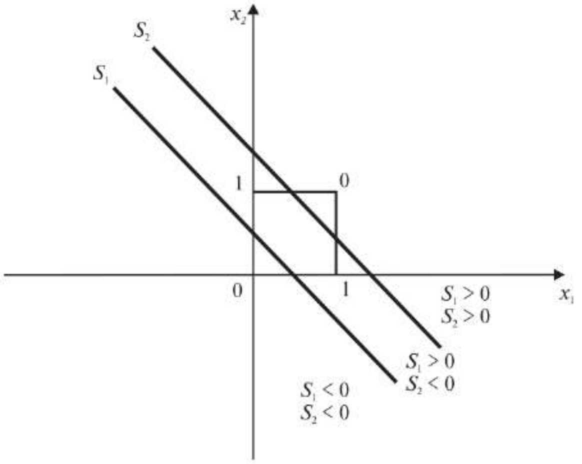


Figure 3. Solving the “EXCLUSIVE OR” problem [11]

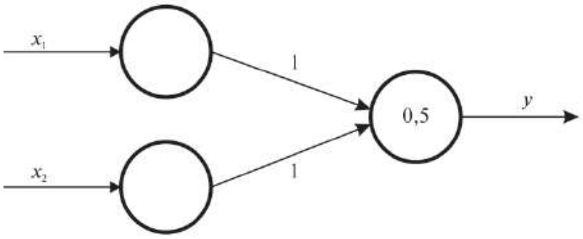


Figure 4. A scheme of a single-layer perseptron solving the “EXCLUSIVE OR” problem [11]

The specification of agents corresponding to the specified actions sets an agent-oriented model for interpreting artificial neural networks. The implementation of this model will be called an artificial neural network interpreter.

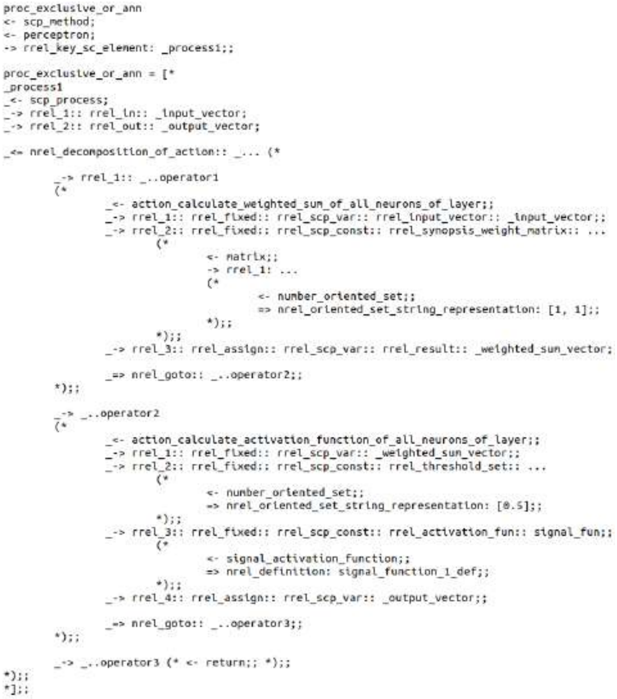
Let us consider an example of a description in the neural network method representation language in SCP, that solves the problem, which is formulated as follows: calculate the result of the “EXCLUSIVE OR” logical operation for the values of two logical variables. In Figure 3, the solution to this problem using a signal function is shown.

In the work [11], a single-layer perseptron that solves the problem is described. The perseptron consists of two input neurons and one output neuron, with a given threshold of 0.5 and a signal activation function:

$$F(S) = \begin{cases} 1, 0 < S < 0, \\ 0, \ else \end{cases}$$

The weight coefficients of the input layer synapses are equal to 1. In Figure 4, a scheme of the perseptron is demonstrated.

This perseptron corresponds to the method represented in the ostis-system knowledge base in the neural network



```
proc_exclusive_or_ann
<- scp_method;
<- perceptron;
-> rrel_key_sc_element: _process1;;

proc_exclusive_or_ann = [*
_process1
_<- scp_process;
_-> rrel_1:: rrel_in:: _input_vector;
_-> rrel_2:: rrel_out:: _output_vector;

_<= nrel_decomposition_of_action:: _... (*

        _-> rrel_1:: _..operator1
        (*
                _<- action_calculate_weighted_sum_of_all_neurons_of_layer;;
                _-> rrel_1:: rrel_fixed:: rrel_scp_var:: rrel_input_vector:: _input_vector;;
                _-> rrel_2:: rrel_fixed:: rrel_scp_const:: rrel_synopsis_weight_matrix:: ...
                        (*
                                <- matrix;;
                                -> rrel_1: ...
                                (*
                                        <- number_oriented_set;;
                                        => nrel_oriented_set_string_representation: [1, 1];;
                                *);;
                        *);;
                _-> rrel_3:: rrel_assign:: rrel_scp_var:: rrel_result:: _weighted_sum_vector;

                _=> nrel_goto:: _..operator2;;
        *);;

        _-> _..operator2
        (*
                _<- action_calculate_activation_function_of_all_neurons_of_layer;;
                _-> rrel_1:: rrel_fixed:: rrel_scp_var:: _weighted_sum_vector;;
                _-> rrel_2:: rrel_fixed:: rrel_scp_const:: rrel_threshold_set:: ...
                        (*
                                <- number_oriented_set;;
                                => nrel_oriented_set_string_representation: [0.5];;
                        *);;
                _-> rrel_3:: rrel_fixed:: rrel_scp_const:: rrel_activation_fun:: signal_fun;;
                        (*
                                <- signal_activation_function;;
                                => nrel_definition: signal_function_1_def;;
                        *);;
                _-> rrel_4:: rrel_assign:: rrel_scp_var:: _output_vector;;

                _=> nrel_goto:: _..operator3;;
        *);;

        _-> _..operator3 (* <- return;; *);;
*);;
*];;
```

Figure 5. A method that solves the "EXCLUSIVE OR" problem represented in the neural network method representation language in SCP

method representation language in SCP. This method is represented in Figure 5.

The description of the method consists of a sequence of two generalized action specifications – action for calculating the weighted sum of all neurons of the layer and action for calculating the activation function of all neurons of the layer.

The signal activation function used in the perseptron is defined in the ostis-system memory by the logic formula shown in Figure 6.

Any agent interpreting actions with arguments given with the *activation function'* relation must use an interpreter of mathematical functions that can be used as activation functions. A classification of such functions is shown in [10].

V. Intelligent framework for building neural network methods

The presence of a language for representing neural network methods and its interpreter in SCP allows for the interpretation of the neural network method in the ostis-system memory. The presence in a common memory of not only instances of methods but also concepts that describe them, creates the basis for automating the process of building (designing and training) neural network methods. The ostis-system memory stores knowledge about the methods of which class can solve the problem of a given class, but instances of this method class may not be represented in the system. In this case, the system should be able to inform the user about the possibility of a solution, for which, however, it is necessary to load a certain method into the system. Since the system stores the problem and the requirements for the method of solving it in a common memory, it becomes possible to design the necessary method. This requires the presence of a design framework for the methods of the corresponding classes. In the case of the neural network method, we are talking about an intelligent framework for building neural network methods.

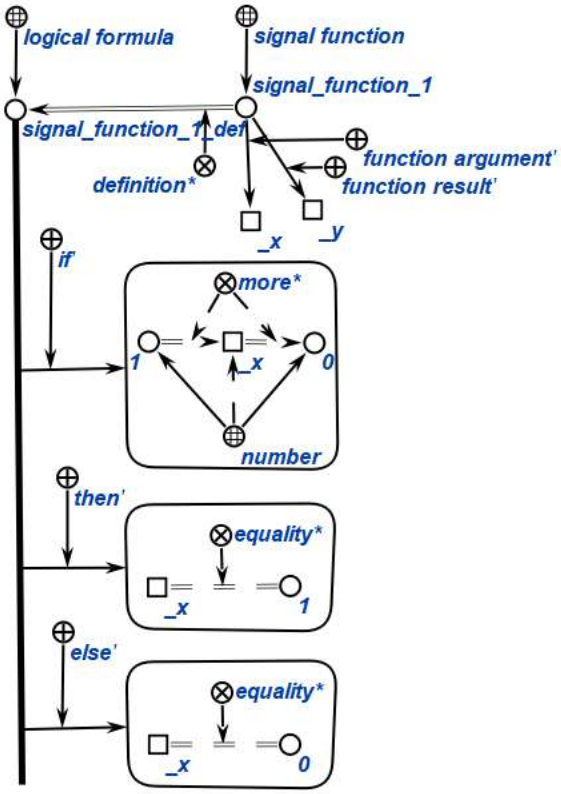


Figure 6. Representation of the activation signal function in the ostis-system memory

The intelligent framework for creating neural network methods is based on corresponding hierarchies of actions, problems, and methods for building ANN. The presence of such a hierarchy will make it possible to describe the method representation language for building ANN and develop an interpreter for that language.

Creation of the hierarchy of the corresponding actions of building ANN should be studied by the stages of design and training of ANN, which, in the general case, are performed by all the developers of ANN:

1. Problem definition.

The problem definition includes a description of the input data (images / video, time series, text), output data, and requirements for the solution method (speed, memory costs, etc.). It also describes additional information that can help in constructing a problem-solving method (for example, the specification of the training dataset, if it exist). Usually, at this stage, the developer determines the class of the problem, forms the requirements for the training dataset, if it is not provided.

The execution of this stage by the ANN design framework involves performing the following actions:

- *Action of problem condition translation*. The action translates the description of the problem specified using the ostis-system interface (for example, natural language interface) into the ostis-system memory. The action is required when the problem condition is specified by the user. It is necessary to understand that the problem description goes into the knowledge base not only from the user interface. For example, a problem can be formulated by the system itself in the duration of its life. This action is common to all ostis-systems, so its consideration goes beyond the consideration of the process of building an ANN intelligent design framework.
- *Action of problem classification*. The action determines the class of the problem (problem of regression, detection, clustering, etc.) based on the description of the problem in the knowledge base.
- *Action of finding a suitable training dataset*. The knowledge base can store a set of dataset specifications to which the ostis-system has access. The action searches for datasets that can be used as a training dataset.
- *Action of generating requirements for the training dataset*. If the training dataset was not provided and was not found, then it is necessary to form a description of the requirements for the training dataset, which can be translated into the user interface language and request the necessary dataset from the user.

2. Dataset preprocessing: cleanup

At this stage, features that have incorrect values are detected (for instance, for some examples, the value of the feature may have an undefined value, or a value that does not match in type, or an abnormally large, or very small value). For features that have an undefined value, various elimination methods can be applied, for example, such values can be replaced by the average value of this feature calculated over all examples (for unsequential data), or they can be replaced by average values from adjacent examples (in the case of sequential data), or some fixed value. A radical method for solving the problem is the removal of examples that have undefined feature values from the dataset. However, it is better to use it if there are few examples with missing feature values. Similar strategies are used for outliers and anomalies (but only if the goal is not to predict these anomalies).

In an intelligent design framework, this stage corresponds to the execution of the *action of dataset cleanup*, which is performed in the case of processing a dataset that was not previously represented in the system memory (for example, was received from the user). The implementation of the interpreter (agent) of this action requires the description in memory of the classification of data cleaning strategies and the implementation of methods for applying these strategies.

3. Dataset preprocessing: identifying meaningful features

Engineering of features is implemented, consisting in the selection of features that affect the output of the model; non-meaningful features that do not correlate with the model output are removed. The purpose of this stage is to reduce the dimensionality of the feature space in order to reduce the influence of the overfitting effect on the model.

To reduce the dimensionality of the feature space, the methods of feature selection and feature extraction can be used.

When selecting features, a subset from the original features is formed (backward selection algorithm, recursive feature elimination algorithm, algorithms using random forests).

When extracting features from a set of features, information is extracted to build a new feature subspace (algorithms using an autoencoder).

In an intelligent design framework, this stage corresponds to the execution of the *action of identifying meaningful features*. The implementation of the interpreter (agent) of this action requires the description in memory of the classification of strategies for reducing the dimensionality of the feature space and the implementation of methods for applying these strategies.

4. Dataset preprocessing: transformation

At this stage, the data is prepared for training. Here, special attention should be paid to the presence of categorical features, most often specified by strings. These features can be nominal and ordinal. To encode ordinal features, a sequential numerical code (1, 2, 3, ...) is most often used. For nominal coding, such a solution is incorrect, since these features are fullright and cannot be compared by a numerical code (for example, gender is 0/1). For nominal features, a direct coding method is used, which consists in creating and using fictitious features according to the number of values of the original one. For example, an attribute of a gender (male, female) is converted into two new features – male and female – with the corresponding values for the existing examples.

Feature scaling involves bringing the feature values to one common interval – this is especially relevant for features that have disproportionate means across all dataset – for example, one feature has an average value of 10.000 and another – 12. This can result in minimizing only by feature with the highest values and poor convergence of the training method. Most often, scaling corresponds to performing normalization on an interval (min-max normalization):

$$x^i_{norm} = \frac{x^i - x_{min}}{x_{max} - x_{min}}$$

where x^i is the value of the feature for a single example i, x_{min} is the smallest value for the feature, x_{max} is the largest value for the feature.

Another scaling technique is to apply feature standardization:

$$x^i_{std} = \frac{x^i - \mu(x)}{\sigma(x)},$$

$\mu(x)$ is a sample mean of a single feature, $\sigma(x)$ is the standard deviation.

Standardization preserves useful information about outliers in the original data and makes the learning algorithm less sensitive to them.

Discretization is used to move from an objective feature to an ordinal one by encoding intervals with a single value (for example, if a feature reflects a person's age, then values can be discretized with the selection of certain age groups, where each group will be encoded by one integer).

In the intelligent design framework, this stage corresponds to the execution of the *action of dataset transformation*. The implementation of the interpreter (agent) of this action requires the description in memory of the classification of methods for scaling features and the implementation of methods for applying these strategies.

5. Dividing the common dataset into training, validation, and test (control) datasets

The entire dataset is divided into training, test, and, in some cases, validation datasets.

The validation set is used to evaluate the impact of changing hyperparameters on the learning outcome and can be used as an additional tool for this along with grid search.

The split is carried out in a ratio of 3:1:1, in percent (60/20/20), if the validation dataset is not used, then 80/20.

In an intelligent design framework, this stage corresponds to the execution of the *action of dataset splitting*.

All previous steps were applied to the dataset; the subsequent steps refer to the used ANN models.

6. Choosing a class of neural network methods in accordance with the formulated problem

At this stage, the selection of the main ANN architecture, which will be used in training, is carried out. However, it should be noted that this selection is relatively conditional; the researcher is not limited to using only one type of ANN to solve a problem (like, for example, a convolutional network for images, since images can also be processed with a conventional multilayer perceptron). Rather, it is about the recommended architecture, but this does not exclude the usage of any other variants of architectures and their combinations within the same model).

Examples of such recommendations are:

- images/video – convolutional neural networks;
- time series – multilayer perceptrons or recurrent networks;
- text information – multilayer perceptrons or recurrent networks;
- sets of characteristics of some objects (for example, car specifications) – multilayer perceptron.

In an intelligent design framework, this stage corresponds to the execution of the *action of selecting a class of neural network methods*.

7. Formation of specifications for input and output data

Additional data transformations are performed related to changing storage structures (for example, converting a multidimensional array to the one-dimensional array, converting types).

In the intelligent design framework, this stage corresponds to the execution of the *action of forming the specification for ANN inputs and outputs*.

8. Selection of optimization method

As part of the work [10], following optimization methods are described:

- stochastic gradient descent (SGD);
- Nesterov method;
- adaptive gradient (AdaGrad);
- adaptive moment estimation (Adam);
- root mean square spread (RMSProp).

In the intelligent design framework, this stage corresponds to the execution of the *action of optimizing method selection*.

9. Selection of an error function to be minimized

At this stage, the error function is set, which will be minimized. For example, MSE is better suited for regression and clustering problems, CE – for classification problems. In the article [10], the classification of such functions within the SD of ANN if described.

These functions are defined as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widetilde{Y_i})^2$$

where n is the size of the dataset, Y_i is the reference value of the function, $\widetilde{Y_i}$ is the output obtained by the NN.

$$CE = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \log(\widetilde{Y_i}) + (1 - Y_i)\log(1 - \widetilde{Y_i}))$$

(case of 2-class classification)

$$CE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{M} Y_i^c \log \widetilde{Y}_i^c$$

(case of multi-class classification)

In the intelligent design framework, this stage corresponds to the execution of the *action of selecting the error function to be minimized.*

10. Initialization of neural network parameters

In the work [10], within the SD of ANN, methods of primary initialization of ANN have already been described. The most commonly used options for initializing neural network weights and thresholds include:

- initialization with values from a uniform distribution over some small interval, for example, [-0.1, 0.1];
- initialization with values from the standard normal distribution;
- Xavier initialization [12].

  It is used to prevent a sharp decrease or increase in the output values of neurons after applying the activation function during the direct passage of the image through a deep neural network. In fact, initialization by this method is carried out by choosing values from a uniform distribution on the interval $[-\sqrt{6}/\sqrt{n_i + n_{i+1}}, \sqrt{6}/\sqrt{n_i + n_{i+1}}]$, where n_i is the number of incoming connections to this layer and n_i is the number of outgoing connections from this layer. Thus, initialization by this method is carried out for different layers of the neural network from different intervals.
- Initialization obtained from the pre-trained model.

  An initialization option that involves using a pre-trained model as a "starting" model, taken from some repository of pre-trained models, trained by the researcher or during the work of an intelligent system.
- Kaiming initialization [13].

  This initialization method is used to solve the problem of "vanishing" gradient and "exploding" gradient. It is performed by selecting values from a uniform distribution on the interval $[-\sqrt{2}/\sqrt{(1+a^2)fan}, \sqrt{2}/\sqrt{(1+a^2)fan}]$, where a is the angle of inclination to the abscissa for the negative part of the ReLU-type activation function (for a common ReLU function, this parameter is 0), fan is the operating mode parameter, which for the forward propagation phase is equal to the number of incoming connections (to eliminate the effect of the "exploding" gradient), and for the backward propagation phase, to the number of outgoing ones (to eliminate the effect of the "vanishing" gradient).

In an intelligent design framework, this stage corresponds to the execution of the *action of initializing ANN.*

11. Selection of ANN hyperparameters

In practice, some hyperparameters (such as the number of layers, their types, the number of neurons in a layer) are often determined experimentally in the process of iterative search for the best solution to the problem. Although there are ways to partially automate this process, they are still designed for the presence of some preconditions for conducting an experiment, in particular, intervals for changing a parameter (for example, learning rate).

Hyperparameters selected at this stage include:

- ANN training parameters (learning rate, momentum parameter, mini-batch size);
- ANN model architecture that is based on previously formulated specifications of input and output data (for example, the number of neurons in a particular layer(-s) or configurations of entire layers).

Finding the optimal hyperparameters can be obtained, for example, using the grid search method, which allows checking the hyperparameter values taken with a certain step or from a certain interval (tuple). Using this method, the optimal set of hyperparameters is selected, which gives the best results; it is used for subsequent additional training. Otherwise, if the results obtained are acceptable, the further learning process is not carried out at all. The cost of this method should be noted, since, in fact, the searching for different values of training parameters is carried out. To reduce the amount of work, a random search method is used.

To optimize the architecture, the types of layers of the neural network, the number of neurons in each layer, their characteristics are determined – the activation function, for convolutional elements – the size of the kernel, as well as the padding parameter and the convolution step (stride). Here, not only the user version of the network can be evaluated but also the pre-trained architecture. The main rule when selecting is that the number of model parameters should not exceed the size of the training dataset. For pre-trained architectures, this restriction is removed.

In an intelligent design framework, this stage corresponds to the execution of the *action of ANN hyperparameter selection.* The action uses the classification and specification of ANN hyperparameters (described within the SD of ANN [10]).

12. Training the model on the dataset

The model is trained until the selected accuracy is achieved (evaluated on the test dataset) or according to other specified criteria (achievement of the specified number of training epochs, invariability of accuracy over the specified number of epochs, drop in accuracy on the validation dataset, etc.).

Training algorithms have already been described in the SD of ANN [10]. Let us demonstrate their classification:

method of training ANN
⊂ *method*
⊃ *method of training with a teacher*
  ⇒ *explanation**:
    [***method of training with a teacher*** is a method of training using the set target variables **]**

⊃ *method of backward propagation of errors*

:= [MBPE]

⇒ *explanation**:

[MBPE uses a certain optimization method and a certain loss function to implement the phase of backward propagation of the error and change the configurable ANN parameters. One of the most common optimization methods is the method of stochastic gradient descent.]

⇒ *explanation**:

[It should also be noted that despite the fact that the method is classified as one of the methods of training with a teacher, in the case of using MBPE for training autoencoders, in classical publications, it is considered as a method of training without a teacher, since in this case there is no marked data.]

⊃ *method of training without a teacher*

⇒ *explanation**:

[***method of training without a teacher*** is the method of training without using the set target variables (in the self-organization mode)]

⇒ *explanation**:

[When performing the algorithm of the method of training without a teacher, useful structural properties of the set are revealed. Informally, it is understood as a method for extracting information from a distribution, the dataset for which was not manually annotated by a human [14].]

In the intelligent design framework, this stage corresponds to the execution of the *action of training ANN*. An example of formalization of this action is shown in Figure 7

13. Evaluating the ANN effectiveness

After training, the resulting model is evaluated using quality assessment metrics.

Further, the result of the evaluation can be visualized with the confusion matrix and the ROC-curve.

The confusion matrix is a matrix (Fig. 8) that contains information about the number of true positive, true negative, false positive, and false negative classifier predictions.

The ROC-curve is a graph in which, based on the given threshold of the classifier solution, the shares of false positives and true positives are calculated. Based on the ROC-curve, the AUC-indicator (area under the curve) is calculated, which is used as a characteristic of model quality.

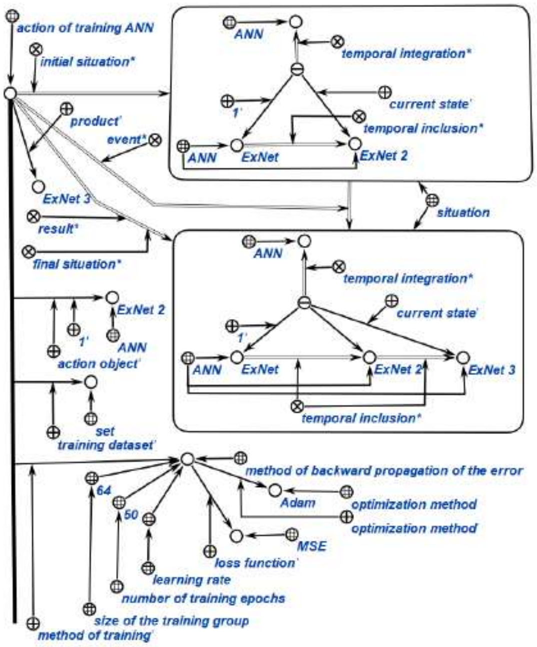
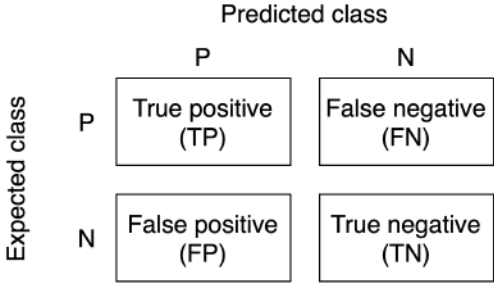


Figure 7. An example of the formalization of the action for training the artificial neural network in the knowledge base [10]



		Predicted class	
		P	N
Expected class	P	True positive (TP)	False negative (FN)
	N	False positive (FP)	True negative (TN)

Figure 8. A confusion matrix

In the intelligent design framework, this stage corresponds to the execution of the *action of ANN performance evaluation*.

Let us consider an example of performance of the described stages by a developer for a specific problem – *classification of digits from the MNIST dataset of handwritten digits*:

1. The initial data of the problem is: a dataset of 70.000 images, pre-divided into a training (60.000 images) and test (10.000 images) datasets. Each image is represented by a two-dimensional array of 28X28 items from the range [0, 255], the numbers represent a shade of gray. In addition, each image has a class label corresponding to a specific digit from 0 to 9.

The problem is: *train a model that will take a two-dimensional array of data as input and return a class* *label corresponding to the recognized digit.*

Thus, the type of problem to be solved is **classification**, the nature of the problem data is **images**.

2. There are no anomalies, erroneous data, or features with missing values in this dataset.

3. In the dataset, there are no non-content features.

4. As a method of data preprocessing we use features scaling, min-max normalization on the interval [0, 1].

5. Let us perform the partition of the train dataset into train and validation datasets at a ratio of 4:1 (48.000 examples in the train dataset and 12.000 examples in the validation dataset).

6. Since the dataset includes images, we will use a convolutional neural network.

7. Formation of specifications for input and output data is not required.

8. We will use the stochastic gradient descent (SGD) method as the optimization algorithm.

9. Since the classification problem is being solved, let us select the cross-entropy loss function as the minimizing function.

10. We will use the Kaiming initialization for the network parameters initialization.

11. In stage 6, it was determined that a convolutional neural network would be used to solve the problem. When using one-hot coding, the last full-connected layer will have 10 neurons according to the number of classes in the problem.

For simplicity, we will use the architecture shown in Fig. 9, which does not contain intermediate layers.

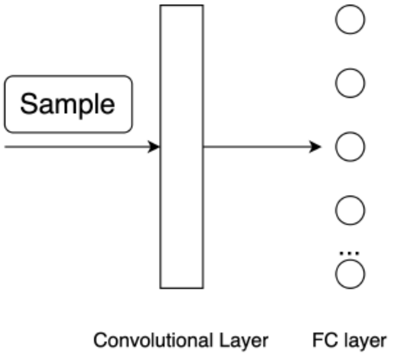


Figure 9. ANN architecture for solving the problem of digit classification

To find the optimal set of hyperparameters, we will apply a random search method.

Let us list the tuples from which hyperparameters will be sampled:

- learning rate – (0.9, 0.1, 0.01, 0.001);
- number of neurons in the convolutional layer – (5, 10, 15, 20);
- size of the convolutional kernel – (3, 5, 7, 9);
- momentum parameter – (0, 0.5, 0.9);
- mini-batch size – (16, 32, 64, 128).

After determining these parameters and evaluating the effectiveness of the algorithm, we obtain the following table:

Table I
PROBLEM RESULTS
(ABBREVIATIONS USED: MBS – MINI-BATCH SIZE, KS – KERNEL SIZE, LR – LEARNING RATE, CNC – CONVOLUTIONAL NEURONS COUNT, ACC – ACCURACY, IT – ITERATIONS COUNT)

#	mbs	ks	lr	momentum	cnc	acc	it
1	128	3	0.001	0.5	10	0.9033	10
2	64	9	0.9	0	15	0.1039	1
3	32	3	0.01	0.5	20	0.9741	10
4	32	7	0.01	0.5	15	0.9794	10
5	16	9	0.001	0.5	20	0.9189	2
6	64	3	0.1	0.5	10	0.9736	10
7	64	7	0.001	0.9	15	0.9007	1
8	32	9	0.1	0.5	5	0.9806	10
9	128	5	0.1	0.5	20	0.98	10
10	32	9	0.01	0.9	5	0.9806	10
11	128	3	0.001	0.9	10	0.893	1
12	32	5	0.9	0.9	20	0.1008	1
13	16	9	0.9	0.5	20	0.0976	1
14	32	7	0.9	0.9	15	0.0932	1
15	128	5	0.01	0.5	20	0.9197	2
16	16	3	0.001	0.5	10	0.904	1
17	16	9	0.001	0	20	0.8866	1
18	128	9	0.1	0.5	5	0.9793	10
19	128	3	0.001	0	10	0.6697	1
20	16	3	0.1	0	15	0.9729	4
21	32	7	0.9	0.5	15	0.1048	1
22	128	7	0.9	0	15	0.1113	1
23	64	9	0.01	0.5	10	0.9482	2
24	16	7	0.9	0	20	0.0985	1
25	16	3	0.1	0.5	5	0.9558	2
26	64	7	0.01	0.9	15	0.9839	10
27	16	7	0.1	0	10	0.9836	10
28	16	5	0.01	0	20	0.9608	2
29	16	5	0.01	0.9	20	0.9847	10
30	32	5	0.01	0.5	15	0.9532	2

It can be noted that the best result (acc = 0.9839) for generalization ability in the validation dataset was obtained with the following parameters: mbs = 64, ks = 7, lr = 0.01, momentum = 0.9, cnc = 15.

12. As a stopping criterion, we selected the simplest one on reaching a given number of epochs of training. No pre-training was performed, and the model obtained after the hyperparameter fitting procedure was used to estimate the generalization ability. The generalization ability on the test dataset was **0.9853**, i.e. **98.53%**.

13. By constructing a confusion matrix based on the trained model and the test dataset, we obtain the result illustrated in Fig. 10

The obtained matrix is diagonally dominant, so the resulting model does relatively few errors.

Based on the analysis of the stages of constructing the ANN that developers perform, the following classification of actions for the construction of ANN can be derived:

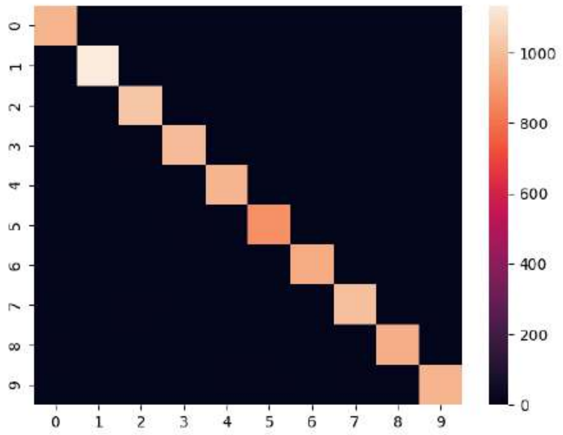


Figure 10. A confusion matrix for the MNIST problem

action of building ANN

⇒ decomposition*:

- {• *action of dataset processing*
  ⇒ decomposition*:
  - {• *action of finding a suitable training dataset*
  - • *action of generating requirements for the training dataset*
  - • *action of dataset cleanup*
  - • *action of identifying meaningful features*
  - • *action of dataset transformation*
  - • *action of dataset splitting*

  }
- • *action of designing ANN*
  ⇒ decomposition*:
  - {• *action of selecting a class of neural network methods*
  - • *action of forming the specification for ANN inputs and outputs*

  }
- • *action of training ANN*
  ⇒ decomposition*:
  - {• *action of optimizing method selection*
  - • *action of selecting the error function to be minimized*
  - • *action of initializing ANN*
  - • *action of ANN hyperparameter selection*
  - • *action of ANN performance evaluation*

  }

}

The implementation of the interpreter of actions for building ANN considered in this section and the description in the knowledge base of the expert knowledge of the ANN developers (and thus the implementation of the intelligent ANN design framework) will automatically, based on the problem description, generate neural network methods in the ostis-system memory, which is one of the key directions for development of this work.

VI. Conclusion

In this article, an approach to the integration and convergence of artificial neural networks with knowledge bases in next-generation intelligent computer systems through the representation and interpretation of the artificial neural network in the knowledge base is described.

The syntax, denotational, and operational semantics of the neural network methods representation language are described, which allows representing and interpreting any ANN in the memory of the intelligent system. The existence of such language generates semantic compatibility of neural network method with other methods represented in the system memory, which allows analyzing the ANN itself and its performance stages by any other methods of the system.

The availability of neural network representation language allows describing the expert knowledge of the developers of the information network in the system memory. In this article, the stages of building ANN, which are carried out by the developers of ANN, are represented. Based on these stages, in order to design an intelligent framework for building neural network methods, the actions of building the neural network methods has been classified and described in the knowledge base.

The design and implementation of the intelligent framework for building ANN in the knowledge base of the system is one of two main directions for further development of this work.

The second main direction is to develop an approach to the processing of fragments of the knowledge base by ANN. For this direction, it is necessary to develop a universal algorithm of mutual-ambiguous matching of knowledge base fragments and input vectors of ANN. A knowledge representation language is able to represent any knowledge. The presence of a neural network method in the system, which is able to take knowledge fragments on the input, will allow solving new, poorly studied classes of problems.

Acknowledgment

The authors would like to thank the research groups of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics and the Brest State Technical University for their help in the work and valuable comments, in particular, Vladimir Golenkov and Daniil Shunkevich.

References

[1] V. V. Golenkov, N. A. Gulyakina, D. V. Shunkevich, *Open technology for ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, G. V.V., Ed. Minsk: Bestprint, 2021.

[2] D. Castelvecchi, "Can we open the black box of AI?" *Nature News*, vol. 538, no. 7623, Oct 2016.

[3] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?: Explaining the Predictions of Any Classifier," 2016. [Online]. Available: https://arxiv.org/abs/1602.04938

[4] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[5] T. R. Besold, A. d'Avila Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kuehnberger, L. C. Lamb, D. Lowd, P. M. V. Lima, L. de Penning, G. Pinkas, H. Poon, and G. Zaverucha, "Neural-symbolic learning and reasoning: A survey and interpretation," Nov. 2017, (accessed 2020, Jun). [Online]. Available: https://arxiv.org/pdf/1711.03902.pdf

[6] A. d'Avila Garcez, T. R. Besold, L. de Raedt, P. Földiak, P. Hitzler, T. Icard, K.-U. Kühnberger, L. C. Lamb, R. Miikkulainen, and D. L. Silver, "Neuralsymbolic learning and reasoning: Contributions and challenges," *In: McCallum, A., Gabrilovich, E., Guha, R., Murphy, K. (eds.) Proceedings of the AAAI 2015 Propositional Rule Extraction under Background Knowledge 11 Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches. AAAI Press Technical Report*, vol. SS-15-03, 2015.

[7] A. Kroshchanka, V. Golovko, E. Mikhno, M. Kovalev, V. Zahariev, and A. Zagorskij, "A Neural-Symbolic Approach to Computer Vision," in *Open Semantic Technologies for Intelligent Systems*, V. Golenkov, V. Krasnoproshin, V. Golovko, and D. Shunkevich, Eds. Cham: Springer International Publishing, 2022, pp. 282–309.

[8] V. Golenkov, N. Gulyakina, I. Davydenko, and D. Shunkevich, "Semanticheskie tekhnologii proektirovaniya intellektual'nyh sistem i semanticheskie associativnye komp'yutery [semantic technologies of intelligent systems design and semantic associative computers]," *Open semantic technologies for intelligent systems*, pp. 42–50, 2019.

[9] D. Shunkevich, "Ontology-based design of hybrid problem solvers," in *Open Semantic Technologies for Intelligent Systems*, V. Golenkov, V. Krasnoproshin, V. Golovko, and D. Shunkevich, Eds. Cham: Springer International Publishing, 2022, pp. 101–131.

[10] M. Kovalev, "Ontology-Based Representation of an Artificial Neural Networks," in *Open Semantic Technologies for Intelligent Systems*, V. Golenkov, V. Krasnoproshin, V. Golovko, and D. Shunkevich, Eds. Cham: Springer International Publishing, 2022, pp. 132–151.

[11] V. A. Golovko and V. V. Krasnoproshin, *Nejrosetevye tekhnologii obrabotki dannyh*. Minsk : Publishing House of the BSU, 2017.

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015. [Online]. Available: https://arxiv.org/abs/1502.01852

[14] J. Goodfellow, I. Benjio, and A. Courville, *Deep learning*. Moscow : DMK Press, 2017.

Конвергенция и интеграция искусственных нейронных сетей с базами знаний в интеллектуальных компьютерных системах нового поколения

Ковалёв М.В., Крощенко А.А., Головко В.А.

В статье рассмотрен подход к интеграции и конвергенции искусственных нейронных сетей с базами знаний в интеллектуальных компьютерных системах нового поколения с помощью представления и интерпретации искусственных нейронных сетей в базе знаний. Описаны синтаксис, денотационная и операционная семантика языка представления нейросетевых методов в базах знаний. Описаны этапы построения нейросетевых методов решения задач с помощью интеллектуальной среды проектирования искусственных нейронных сетей.

Received 14.11.2022

Deep neural networks application in next-generation intelligent computer systems

Aliaksandr Kroshchanka
Brest State Technical University
Brest, Belarus
kroschenko@gmail.com

***Abstract*—In the article, an approach to building hybrid next-generation intelligent computer systems (NGICS) based on the integration of pre-trained models of deep neural networks and logical models developed using the OSTIS technology is proposed. To reduce the requirements for the size of the training dataset, the authors propose an alternative method for pre-training deep models. To achieve the interpretability of neural network models, the authors used methods from the Explainable AI (XAI) field.**

***Keywords*—Neuro-symbolic approach, OSTIS, deep neural networks, Explainable AI, SHAP, hybrid intelligent systems**

I. Introduction

The implementation of next-generation intelligent computer systems is one of the most promising objectives of the present and near future of AI science. New approaches built at the intersection of various directions of artificial intelligence allow eliminating or minimizing the impact of the shortcomings of individual methods, while enhancing the overall efficiency of intelligent systems. For example, when combining the capabilities of logical and neural network models in the context of implementing a neuro-symbolic approach in AI, we benefit from each model used. From logical models – the possibility of explaining the results for a user of the system who is not expert in the subject domain of the problem, from neural networks – the possibility of solving problems that are difficult to formalize (for example, data analysis and computer vision) [1]. Well-known scientists and researchers in certain fields of AI are increasingly declaring the need for compatibility between logical and neural network approaches (for example, [2]).

It should be noted that progress has already been made in these separate fields of research. For example, thanks to logical approaches in artificial intelligence (for example, an OSTIS technology [3]), systems for automating the work of manufacturing enterprises [4] are being developed. On the other hand, in the last decade, there has been a tendency towards the active usage of machine learning methods (and neural networks) to solve various problems. Thanks to the development of deep learning theory, new approaches and models have solved problems, that earlier have been solved successfully only by humans, and in some cases even outperform them (for example, [5]). Such tendencies definitely give grounds to further actively explore neural network models, applying these approaches in new, still poorly explored or high-cost fields.

The organization of the neural network training process is the cornerstone of the achievements obtained using these models. It should be noted that most of the research work currently is based on the usage of so-called pre-trained neural networks. These are networks that have already been trained, and they have been retrained for a new problem being solved (transfer learning [6]). Thus, model training is put on stream, making the threshold for entering the field lower than ever before. However, this potentially leads to a serious commercialization of the field of neural networks with the impossibility (primarily via the hardware lack) for ordinary researchers to train neural networks from the scratch. In addition, not in all cases, the usage of pre-trained models and transfer learning can help in solving new problems [7].

These circumstances form the need to develop new methods for training deep neural networks that reduce the requirements for hardware and the size of used dataset.

Despite the success of neural network models, there remains a certain caution in their usage, mainly due to the closed nature and non-interpretability of these models. We see a solution to this problem in the development of hybrid approaches.

In hybridization, a collision with the problem of integrating different models is inevitable. In the analysis of neural network models it is necessary to proceed from the output data generated by the model. The direct usage of the output data in the development of hybrid intelligent systems is possible and gives results that allow talking about their effectiveness [8]. However, in this case, the neural network model is used in the "black box" mode, without the possibility of interpreting the impact that the input data has on the final result. A successfully implemented interpretation would allow the integration of models at a qualitatively new level, supplementing the logical subsystem with new rules based on the identified patterns. In this case, the neural network part could be used in the process of training an intelligent system to form rules, and then, if necessary, "disable" and generate predictions only based on logical rules. On the other hand, it is possible to use models in combination, getting the primary result from the neural network subsystem, and then form an interpretation for the user, which would allow leveling the wariness in using neural networks. In addition, it becomes possible to obtain information that allows more accurately assessing the quality of the model in the face of possible leaks and shifts in the data.

The authors of this article propose to implement such integration using the results of research obtained in the field of Explainable AI (XAI), which make it possible to assess the influence of the values of the features on outputs of the neural network. Such an evaluation can be represented as a sorted list of features that most strongly influence the result, as well as the formation of feature change intervals, within which the features make the greatest contribution to the results obtained at the output of the neural network.

Within this article, a solution to the problem of integrating neural networks and the OSTIS technology based on Explainable AI and deep learning methods is proposed.

The next sections are organized as follows: Section II describes the problem definition for the development of a next-generation intelligent computer system; Section III provides an overview of existing and proposed approaches in the field of training deep neural networks; Section IV gives a brief description of the SHAP method; the main practical results obtained by the authors are represented in Section V; Section VI summarizes the main conclusions of the proposed approach and describes the possibilities for its evolution.

II. Problem definition

Based on the abilities of next-generation intelligent systems, which are given in [3], we formulated a set of basic requirements for such systems:

- semantic compatibility with existing approaches in the field of artificial intelligence;
- evolution of the system;
- replaceability of system components – the ability to replace the active components of the system directly in the process of system work, for example, in the process of selecting a solution to a given problem;
- (self-)extensibility, simplicity in making changes to the existing set of components with the complication of their functionality;
- no side effects when using the system;
- ability to explain decisions;
- adaptive interface.

Semantic compatibility primarily refers to the possibility of using various AI technologies as system components without the need for constant redesign of the system in the face of changing theories and requirements.

The core of the next-generation system is the concept of compatibility of approaches in the field of AI. Fundamentally, this means the possibility of coexistence of approaches developed in different fields of science within the same system. For example, logical and fuzzy models or logical and neural network models, etc. At the same time, a fundamental boundary should be drawn between systems (or approaches on the basis of which intelligent systems were implemented) of the previous generation, where developers used hybrid methods – for example, neural networks, which can act as separate individuals of a population and thus participate in the implementation of a certain genetic algorithm. In such cases, it was about hybrid methods, but such methods did not include components that provide the necessary level of reflection of the intelligent system. Next-generation intelligent systems are able to explain the decisions they make. For such systems, one of the main requirements is their **evolution**, i.e. the ability to change not only their state but their qualities, which is the most valuable property. Given the presence of semantic compatibility, such a system is able to do this in the most natural way, and **replacement of components**, even if they have different nature but solving the same problem, is not difficult.

Extensibility in the presence of these properties is only a matter of developer competence. The **self-extensibility** of the system becomes the development of extensibility, which is represented in the possibility of generating its own components by the system based on the available knowledge. This part is the most valuable and even revolutionary, since the components offered by the system itself are unique ways to solve problems, as well as ones that may not have been known until now.

In addition to the listed properties, the system must be **free of side effects**, that is, it must not do anything for which it was not designed and intended. This functionality can be considered in the context of some “stop tap”, a set of directives of direct and unconditional action that determine the purpose setting of the system and its internal value system.

A property directly related to the previous one is the **ability to explain decisions**. Since the NGICS evolves and acquires the ability to create its own components, the correct interpretation of the obtained decisions is very important. Based on the purposes and directives available in the system, the system should describe the procedure for making decisions with the explaining for each step. Thus, the components created by the system will be documented by the system itself.

Finally, an **adaptive interface** forms a convenient user access to the NGICS. Such interface is not limited to adaptation to the user device through which the system is accessed but also to adaptation to the users themselves, to their physical abilities and limitations, habits and interests. Only such an interface can be called fully adaptive.

All these requirements must be taken into account when creating a model of NGICS.

The general view of the proposed model is shown in Fig. 1.

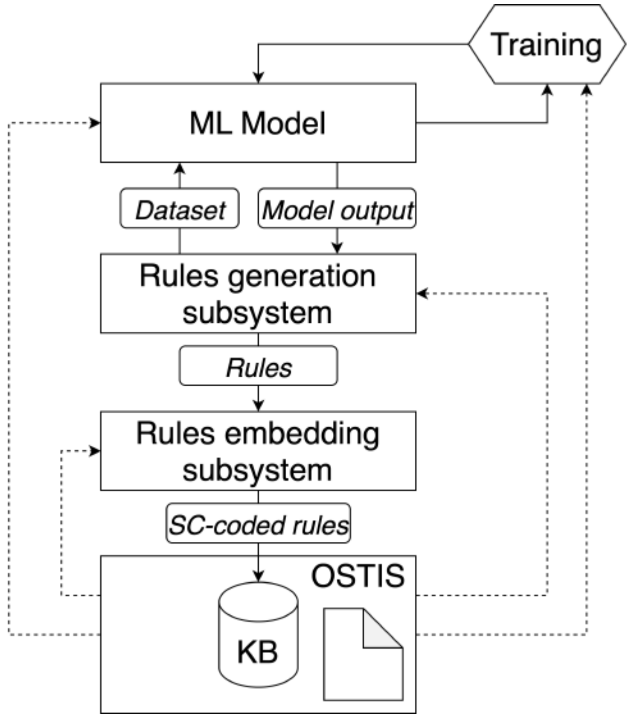


Figure 1. A model of a hybrid system (dashed lines indicate control actions)

Here, it is necessary to emphasize the fact that neural networks are ideally suited as component of next-generation intelligent computer systems. This is achieved mainly by the fact that these models are adaptive and can be used to solve various problems. In addition, such models support additional training during work. As the initial version of the neural network model, a model pre-trained on a small dataset can be used.

It should be noted that the neural network subsystem can also be placed in the OSTIS system and interact with the knowledge base as an agent. In this article, we propose a simplified architecture, focusing on the idea of interpretability of the neural network subsystem.

III. Deep neural networks training

Today, there are two main approaches to train deep neural networks: the first involves pre-training according to Hinton, the second involves special types of activation functions (ReLU), a large available training dataset, and some special regularization techniques (for example, dropout).

At the same time, it is necessary to distinguish between pre-training as a pre-executed procedure in accordance with the Hinton approach based on greedy layer-wise unsupervised learning with Restricted Boltzmann Machine as base trained network (let us call it pre-training of type I – Fig. 2) and pre-training as the process of preparing a pre-trained neural network that can be retrained on a different dataset to solve other problems using transfer learning (pre-training of type II). In the second case, traditional learning techniques can be used (for example, stochastic gradient descent with ReLU activation functions).

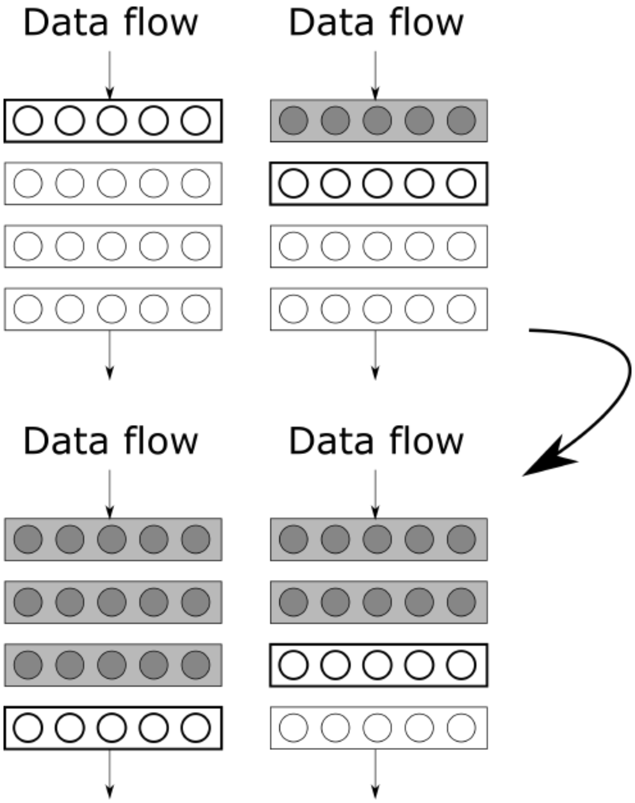


Figure 2. Greedy layer-wise pre-training

The choice of one or another approach to training deep neural networks depends on the size of the training dataset. So, if the dataset is large, pre-training of type II is applied. Otherwise, pre-training of type I is used. For small training datasets, this method overcomes overfitting [9].

The purpose of applying pre-training (both the first and second types) is to achieve some “good” initial initialization of the parameters of the neural network model. This allows starting the retraining process with a lower generalization error and speed it up.

In this article, a variant of the pre-training method based on the Hinton type is used. Further, the pre-training method proposed by Hinton will be called the classical method.

Let us consider a model of a restricted Boltzmann machine.

This model consists of two layers of stochastic binary neurons, which are interconnected by bidirectional symmetrical connections (Fig. 3). The input layer of neurons is called visible layer (X), and the output layer is called hidden layer (Y). The restricted Boltzmann machine can generate any discrete distribution if enough hidden layer neurons are used [10].

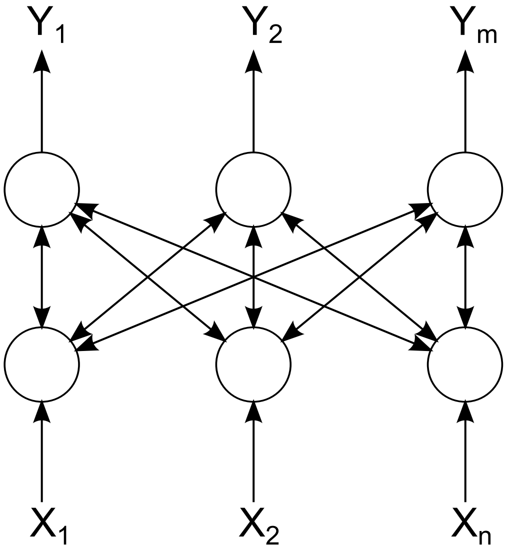


Figure 3. A Restricted Boltzmann Machine

This network is a stochastic neural network in which the states of visible and hidden neurons change in accordance with the probabilistic version of the sigmoid activation function:

$$p(y_j|x) = \frac{1}{1+e^{-S_j}},\ S_j = \sum_i^n w_{ij}x_i + T_j$$

$$p(x_i|y) = \frac{1}{1+e^{-S_i}},\ S_i = \sum_j^m w_{ij}y_j + T_i$$

where w_{ij} are the weight coefficients of the neural network, S_i, S_j are the weighted sums calculated for the neurons of the visible and hidden layers, respectively, T_i, T_j are the thresholds of the visible and hidden layers.

The rules for online learning of a restricted Boltzmann machine proposed in the classical method are as follows [11]:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(1)y_j(1))$$

$$T_i(t+1) = T_i(t) + \alpha(x_i(0) - x_i(1))$$

$$T_j(t+1) = T_j(t) + \alpha(y_j(0) - y_j(1))$$

where $x_i(0), x_i(1)$ are the original data of the visible layer and data, which been restored by the neural network, $y_i(0), y_i(1)$ are the original data of the hidden layer and data, which been restored by the neural network.

The last equations are obtained using the Contrastive Divergence algorithm with the parameter $k = 1$.

Rules for an arbitrary natural k can be obtained similarly:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(k)y_j(k))$$

$$T_i(t+1) = T_i(t) + \alpha(x_i(0) - x_i(k))$$

$$T_j(t+1) = T_j(t) + \alpha(y_j(0) - y_j(k))$$

Rules for batch learning are as follows (case CD-1):

$$w_{ij}(t{+}1) = w_{ij}(t){+}\frac{\alpha}{L}\left(\sum_{l=1}^{L}(x_i^l(0)y_j^l(0){-}x_i^l(1)y_j^l(1))\right)$$

$$T_i(t+1) = T_i(t) + \frac{\alpha}{L}\left(\sum_{l=1}^{L}(x_i(0) - x_i(1))\right)$$

$$T_j(t+1) = T_j(t) + \frac{\alpha}{L}\left(\sum_{l=1}^{L}(y_j(0) - y_j(1))\right)$$

It should be noted that in order to obtain these rules, Hinton was guided by the idea of maximizing the likelihood function of the form:

$$P(x) = \sum_y P(x,y)$$

where $P(x,y)$ is the probability for a case of a visible and hidden neuron in the state (x,y), determined on the basis of the Gibbs distribution $P(x,y) = \frac{e^{-E(x,y)}}{Z}$, $Z = \sum_{x,y} e^{-E(x,y)}$ is the probability normalization parameter, E is the energy of the system in the state (x,y).

Finally, the function will take the form:

$$P(x) = \sum_y \frac{e^{-E(x,y)}}{Z} = \frac{\sum_y e^{-E(x,y)}}{\sum_{x,y} e^{-E(x,y)}}$$

Previously, the authors proposed an approach that generalizes the classical approach and demonstrated its effectiveness for some problems (for example, [12]).

The rules for online learning in accordance with the proposed approach for CD-1 are as follows:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) \\ &- \alpha((y_j(1) - y_j(0))F'(S_j(1))x_i(1) + \\ &(x_i(1) - x_i(0))F'(S_i(1))y_j(0)), \\ T_i(t+1) &= T_i(t) - \alpha(x_i(1) - x_i(0))F'(S_i(1)), \\ T_j(t+1) &= T_j(t) - \alpha(y_j(1) - y_j(0))F'(S_j(1)). \end{aligned}$$

The rules for batch learning in accordance with the proposed approach for CD-1 are as follows:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) \\ &- \frac{\alpha}{L}\Bigg(\sum_{l=1}^{L} \Delta y_j^l(1)x_i^l(1)F'(S_j^l(1)) + \\ &\Delta x_i^l(1)y_j^l(0)F'(S_i^l(1))\Bigg), \end{aligned}$$

$$T_j(t+1) = T_j(t) - \frac{\alpha}{L}\left(\sum_{l=1}^{L} \Delta y_j^l(1) F'(S_j^l(1))\right),$$

$$T_i(t+1) = T_i(t) - \frac{\alpha}{L}\left(\sum_{l=1}^{L} \Delta x_i^l(1) F'(S_i^l(1))\right),$$

where $\Delta y_j^l(1) = y_j^l(1) - y_j^l(0)$, $\Delta x_i^l(1) = x_i^l(1) - x_i^l(0)$

When obtaining these rules, the authors were guided by the idea of minimizing the mean squared error of the network (the case of using CD-1):

$$E_s(1) = \frac{1}{2L}\left(\sum_{l=1}^{L}\sum_{j=1}^{m}(\Delta y_j^l(1))^2 + \sum_{l=1}^{L}\sum_{i=1}^{n}(\Delta x_i^l(1))^2\right)$$

where $\Delta y_j^l(1) = y_j^l(1) - y_j^l(0)$, $\Delta x_i^l(1) = x_i^l(1) - x_i^l(0)$, L – size of the training dataset.

It is possible to prove the identity of these learning rules to the classical ones by using neurons with a linear activation function.

Thus, the following theorem can be proved:

Theorem. Maximizing the likelihood function of data distribution $P(x)$ in the space of synaptic connections of a restricted Boltzmann machine is equivalent to minimizing the mean squared error of the network E_s in the same space using linear neurons.

IV. Interpretability of ANN

The problem of interpretability of machine learning models is currently quite effectively solved by the Explainable AI methods [13].

In XAI methods such as LIME [14] and SHAP [15], only the data feeded to the model and the output returned by the model are used in the analysis. This type of methods belongs to the model-agnostic type, i.e. they can be applied to any machine learning model.

Our hybrid intelligent system model uses the SHAP (SHapley Additive exPlanations) approach to interpret the results obtained by the neural network.

The SHAP method is based on an attempt to explain changes in the predictions of the model caused by a change in the input features or the appearance of some information about the input feature. In this case, the contribution of each feature to the prediction of the model is calculated.

The SHAP method is based on the game theory. The key quantities used in evaluating the contribution of each feature to the overall output of the model are the Shapley values.

In this case, the players are features (the presence of the *i*-th player means the current value of the *i*-th feature in the example *x*, the absence of the *i*-th player means the undefined value of the *i*-th feature), and all the represented features define a set of players, called coalition.

Denote by $f : X \to Y$ the model under study, $x \in X$ is the selected test example for which the output value of the model is interpreted, $X \in \mathbb{R}^N$ is the feature space, N is the number of features (players), ν is a characteristic function that assigns a number to each coalition of players – its efficiency.

Further, assuming that some of the features in the example x are known and some are omitted (have undefined values), we obtain the vector x_S corresponding to the known features.

The Shapley values for each player are calculated using the following formula:

$$\phi(i) = \sum_{S \in 1,2,\ldots,N} \frac{|S|!(|N| - |S| - 1)!}{N!}\Delta(i, S)$$

where S defines the coalition of players and $\Delta(i, S)$ is the efficiency gained from adding player i to the coalition of players S:

$$\Delta(i, S) = \nu(S \cup i) - \nu(S)$$

In the SHAP method, the conditional expectation is used as the characteristic function for the set of features *S* of the example *x*:

$$\nu(S) = E[f(x)|x_S]$$

In practice, when calculating the characteristic function given by the last formula, simplifications are used (for example, Kernel SHAP modification).

V. Experimental results

For the experimental part of the research, we chose the well-known Fisher Irises dataset.

The size of this dataset is 150 examples, which are divided into train (120 examples) and test (30 examples) datasets. The examples describe the geometric shape of the iris flower. Each example contains 4 features (sepal length, sepal width, petal length, petal width) and a class label (0–2). It is required to classify the example according to the type of flower (Iris setosa, Iris virginica, Iris versicolor).

The usage of such simple dataset made it possible, on the one hand, to demonstrate the effect of pre-training on a dataset of a limited size and, on the other hand, to show the process of interpreting the model with the construction of simple rules, for which checking their corectness is not difficult.

For this problem, a series of experiments was carried out with the following training options:

- no pre-training – in this case we used backward propagation to train neural network from the scratch;
- with pre-training by the classical method;
- with REBA-based pre-training.

The training process used a model with a structure (4 – 10 – 10 – 3) with ReLU activation functions on all layers, except for the last one.

Table I
PRE-TRAINING PARAMETERS

mini-batch size	momentum	epochs count	train rate
4	0.5	5	0.01/0.04

Table II
TRAINING PARAMETERS

mini-batch size	momentum	epochs count	train rate
4	0.9	10	0.01

Tables I and II show the main parameters of pre-training and training stages.

A series of 100 computational experiments was carried out, the results of which were averaged. The results are represented in Table III.

Table III
RESULTS

pre-training method	test efficiency, %
RBM	91.0
REBA	92.6
Without pre-training	83.73

During the implementation of the SHAP method, Shapley values were obtained, on the basis of which visualizations were drawn. In Fig. 4, 5, and 6, the cumulative influence of individual features on irises type classification is shown.

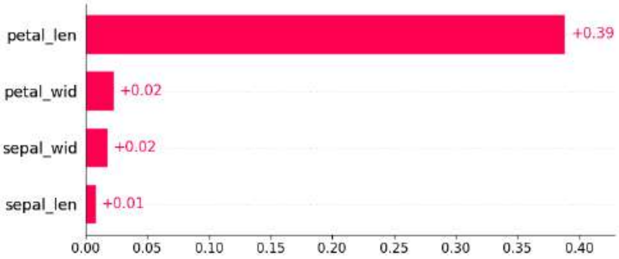


Figure 4. Influence of features on class 0 for identification (Iris setosa)

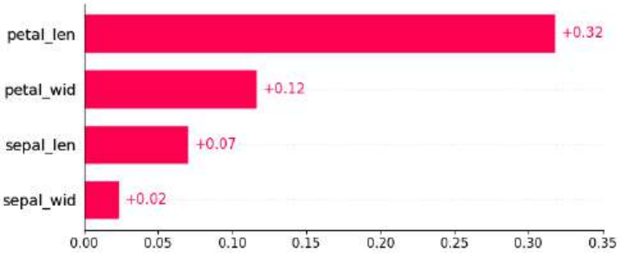


Figure 5. Influence of features on class 1 for identification (Iris virginica)

According to these images, it can be seen that the petal length feature has the greatest influence on determining the type of flower. This is confirmed by a more detailed study of the values dependence from this feature on the Shapley values (Fig. 7, 8, 9).

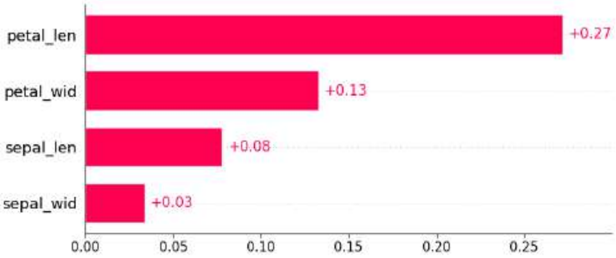


Figure 6. Influence of features on class 2 for identification (Iris versicolor)

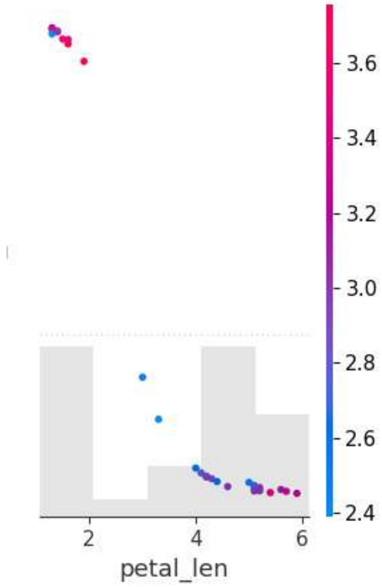


Figure 7. Dependences of feature values on Shapley values (class 0)

As can be seen from the represented visualizations, all values of the **petal length** feature are concentrated in 3 main intervals that directly affect the class identification ([1, 2], [2, 5], [5, 6]). The boundaries of the ranges for simplification are defined approximately. Based on these data, rules can be formulated:

1) If the value of the petal length feature is in the range from [1, 2], then define the class of the flower as **Iris setosa**
2) If the value of the petal length feature is in the range from [2, 5], then define the flower class as **Iris virginica**
3) If the value of the feature petal length is in the range from [5, 6], then define the flower class as **Iris versicolor**

These rules take into account only the value of one feature, in order to improve the characteristics of

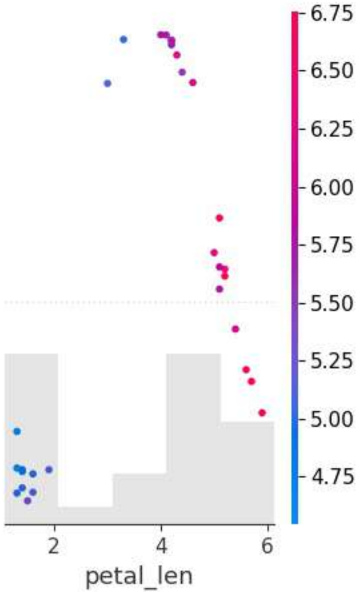


Figure 8. Dependences of feature values on Shapley values (class 1)

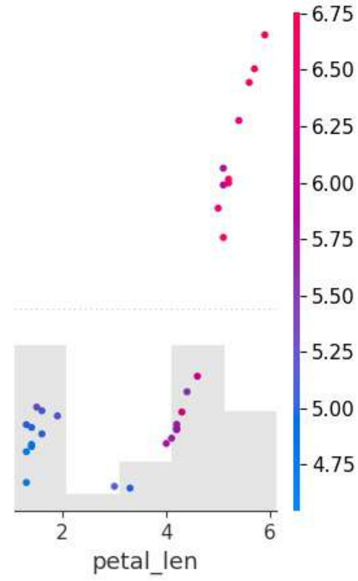


Figure 9. Dependences of feature values on Shapley values (class 2)

the classifying algorithm; the number of rules can be expanded by analyzing changes in other main features.

Finally, after forming a natural language representation of the rules, they can be easily represented in the SC-code or illustrated using its visual representation in SCg (Fig. 10).

VI. Conclusion

In the article, an approach to the implementation of next-generation intelligent computer systems is proposed, which allows integrating neural network and logical models created using the OSTIS technology. The proposed approach is based on the application of deep neural network pre-training methods and Explainable AI. The effectiveness of the proposed approaches to the pre-training of a deep neural network, as well as the approach to integration, is shown on the example of solving classification problem.

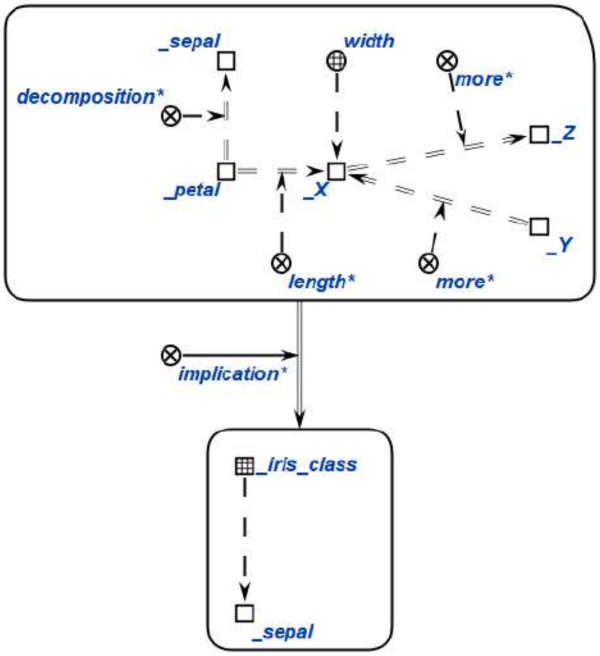


Figure 10. Representation of the rule for determining the type of flower in SCg

This approach can be used in the development of next-generation intelligent computer systems, for which the small amount of available training data and high requirements for the interpretation of the results often become critical factors.

As directions for further work, the authors see the development of the proposed approach in the context of studying the applicability to convolutional models, as well as studying the possibilities of interpreting models with homogeneous inputs (for example, when solving the problem of recognizing objects in an image, where the role of an individual pixel or superpixel is difficult to formalize).

Acknowledgment

The author would like to thank the research groups of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics and the Brest State Technical University for their help in the work and valuable comments, in particular, Vladimir Golovko, Vladimir Golenkov, and Daniil Shunkevich.

The work was supported by the Belarusian Republican Foundation for Fundamental Research BRFFR, project F22KI-046.

References

[1] A. Kroshchanka, V. Golovko, E. Mikhno, M. Kovalev, V. Zahariev, and A. Zagorskij, “A Neural-Symbolic Approach to Computer Vision,” in *Open Semantic Technologies for Intelligent Systems*, V. Golenkov, V. Krasnoproshin, V. Golovko, and D. Shunkevich, Eds. Cham: Springer International Publishing, 2022, pp. 282–309.

[2] Y. Cun, *Quand la machine apprend: La révolution des neurones artificiels et de l’apprentissage profond*. Odile Jacob, 2019. [Online]. Available: https://books.google.by/books?id=78m2DwAAQBAJ

[3] V. V. Golenkov, N. A. Gulyakina, D. V. Shunkevich, *Open technology for ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, G. V.V., Ed. Minsk: Bestprint, 2021.

[4] V. Taberko, D. Ivaniuk, N. Zotov, M. Orlov, O. Pupena, and N. Lutska, “Principles of building a system for automating the activities of a process engineer based on an ontological approach within the framework of the Industry 4.0 concept,” in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*, vol. 5, Minsk, 2021, pp. 209–218.

[5] L. E. van Dyck, R. Kwitt, S. J. Denzler, and W. R. Gruber, “Comparing Object Recognition in Humans and Deep Convolutional Neural Networks—An Eye Tracking Study,” *Frontiers in Neuroscience*, vol. 15, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2021.750639

[6] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, “A comprehensive survey on transfer learning,” 2019. [Online]. Available: https://arxiv.org/abs/1911.02685

[7] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[8] V. Golovko, A. Kroshchanka, M. Kovalev, V. Taberko, and D. Ivaniuk, “Neuro-Symbolic Artificial Intelligence: Application for Control the Quality of Product Labeling,” in *Open Semantic Technologies for Intelligent System*, V. Golenkov, V. Krasnoproshin, V. Golovko, and E. Azarov, Eds. Cham: Springer International Publishing, 2020, pp. 81–101.

[9] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, no. 521 (7553), pp. 436–444, 2015.

[10] Y. Bengio, “Learning deep architectures for ai,” *Foundations and Trends in Machine Learning*, no. 2(1), pp. 1–127, 2009.

[11] G. Hinton, S. Osindero, and Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” *Neural computation*, vol. 18, pp. 1527–54, 08 2006.

[12] V. Golovko, A. Kroshchanka, and E. Mikhno, “Deep Neural Networks: Selected Aspects of Learning and Application,” in *Pattern Recognition and Image Analysis*. Cham: Springer International Publishing, 2021, pp. 132–143.

[13] A. Thampi, *Interpretable AI: Building Explainable Machine Learning Systems*. Manning, 2022. [Online]. Available: https://books.google.by/books?id=ePN0zgEACAAJ

[14] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why Should I Trust You?: Explaining the Predictions of Any Classifier,” 2016. [Online]. Available: https://arxiv.org/abs/1602.04938

[15] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

Применение глубоких нейронных сетей в интеллектуальных компьютерных системах нового поколения

Крощенко А. А.

Статья посвящена модели гибридной интеллектуальной системы нового поколения, базирующейся на интеграции предобученных глубоких нейросетевых моделей и логических моделей технологии OSTIS. Для снижения влияния объема обучающей выборки на процесс обучения модели авторами предлагается альтернативный подход к предобучению глубоких нейронных сетей. Для достижения цели интерпретируемости нейросетей использовались методы из области Explainable AI.

Received 14.11.2022

Automatic Construction of Classifiers by Knowledge Ecosystem Agents

1st Viktor Krasnoproshin
Faculty of Applied Mathematics and Computer Science
Belarussian State University
Minsk, Belarus
krasnoproshin@bsu.by

2nd Vadim Rodchenko
Faculty of Mathematics and Informatics
Yanka Kupala State University of Grodno
Grodno, Belarus
rovar@grsu.by

3rd Anna Karkanitsa
Faculty of Mathematics and Informatics
Yanka Kupala State University of Grodno
Grodno, Belarus
a.karkanica@grsu.by

Abstract—**Digital Ecosystem (DE) is understood as a distributed, adaptive, open socio-technical system that has the properties of self-organization, scalability and sustainability. A high level of adaptability and self-organization of DE can only be ensured by the Knowledge Ecosystem (KE) built on the principle of nesting dolls.**

The main purpose of KE is effective knowledge management. This is achieved as a result of improving the interaction environment for system participants, simplifying the decision-making process and stimulating innovations. The base elements of the Knowledge Ecosystem are software agents. They "live" in the ecosystem environment: receive and analyze data about surrounding events, interpret them and execute commands that affect the environment.

The paper presents the process of automatic construction of classifiers based on the interaction of knowledge ecosystem agents. The initial information for the collaborative work of agents is the alphabet of classes, the a priori dictionary of features (PDF), and the data warehouse. The effectiveness of the proposed approach is demonstrated by the example of processing model data.

Index Terms—**knowledge ecosystem, data mining, multi-agent system, knowledge discovery, instance-based learning, training set**

I. Introduction

The concept of an ecosystem was borrowed from biology and has been developed in the early 21th century. In the classical sense, an ecosystem is an open system characterized by input and output flows of matter and energy. Any ecosystem is a complex self-organizing, self-regulating and self-developing system, which is an integration project with many participants [1].

Digital Ecosystem is a digital space where variety of services of a unified environment operate seamlessly. Such integration allows to manage user behavior, to achieve maximum speed and transparency of processes, to detect issues and identify ways of improvement in different areas of activity [2]. The development and implementation of KE systems is one of the priorities for the information technology development and use.

DE combines various elements of a digital platform. It ensures their interaction and connection with the world around [3]. The base elements of the system are a single account, digital services for solving various problems, server infrastructure, teams of developers and engineers, customers and other stakeholders [4].

Currently, there are three classes of digital ecosystems: functional ecosystem, platform ecosystem and super platform ecosystem [5].

Functional Digital Ecosystems are one of the simplest ecosystems. They are built around an existing product or company offering and are characterized by a relatively small number of participants (companies and partners). They tend to focus on the internal aspect and therefore are often a closed ecosystem.

Platform Ecosystems are characterized by a large number of partners (can be several million) and based on the use of a common platform that all partners use together and create their own value.

The most complex are the Super Platform Ecosystems. They provide the ability to connect and interact with almost any number of partners. In this case, integration is provided not only at the level of services, but also between platforms.

Inside the DE, according to the principle of a nesting doll, a knowledge ecosystem (KE) is placed. KE is an adaptive system that includes a database, a knowledge base, and intelligent agents [6]. The base components of KE are the technological core, critical interdependencies, knowledge agents and performative actions [7].

The goal of the knowledge ecosystem is the effective implementation of the decision-making process through high-quality interaction between its agents and components [8]. Being inside KE, knowledge agents receive data about ongoing events, interpret them, and execute commands that affect the environment. Agents have such important properties as autonomy, social ability, reactivity and pro-activity [9].

The paper presents an original method of automatic construction of classifier based on the interaction of knowledge ecosystem agents. The results of its practical application on model data for solving the classification problem are presented.

II. Multi-Agent Interaction

At present, working out and use of the concept of multi-agent systems (MAS) is one of the priority direction of information technology development.

There are a number of requirements for the knowledge management mechanism in the KE. In particular, this mechanism should ensure both the development of interactions between the exchange participants and the simplification of the decision-making process, as well as driving innovation due to the evolution of cooperation between agents.

An alternative to directive management methods in KE are strategies that are based on self-organization (as a response to external changes).

The development of multi-agent systems technology has led to a change in the requirements for agents. Initially, an intelligent agent was considered as a powerful subsystem that had to have a global vision of the problem and have all the necessary abilities, knowledge and resources to solve it [10].

In recent years (in the field of MAS) an approach that focuses on the narrow specialization of agents has become a priority. Each agent must provide a solution to some problem. The solution of a complex problem is based on the agent's interaction within the MAS (Fig. 1).

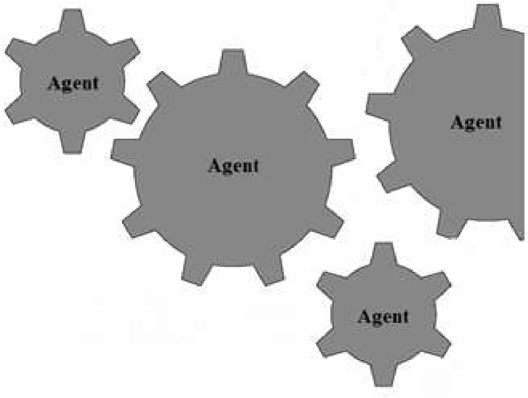


Figure 1. Multi-agent interaction.

Each agent is a complex object that can manipulate other objects and has advanced tools of interacting with the environment and its own kind. A single agent is a software/hardware implemented system and has the following characteristics:

- *autonomy or semi-autonomy*: functioning without outside interference and ensuring self-control over one's actions and internal states;

- *social ability*: interaction with other agents by exchanging messages using communication tools;

- *reactivity*: the ability to perceive the state of the external environment;

- *pro-activity*: the response of agents not only to incentives coming from the environment, but also a goal-directed manifestation of the initiative [11].

III. Classifier Construction Based on the Interaction of Three Agents

At present, one of the trends of information systems development is the development and implementation of intelligent digital ecosystems.

Currently, the progress on using artificial intelligence technologies is largely provided by machine learning methods. The essence of these methods is related to the identification of empirical patterns in the data. During the learning process sets of positive and negative examples (related by an unknown pattern) are analyzed and a classification algorithm is developed (providing the separation of examples into two classes).

In fact, machine learning provides for the construction of decision rules that implicitly express empirical patterns. For instance, as a result of *Supervised Learning* a *classifier* is built, which is a *"black box"* since it cannot be interpreted in terms of the subject domain.

Knowledge discovery is the process of discovering previously unknown, useful and interpretable patterns in the initial data sets required for effective decision-making [12].

If it is possible to discover the distinctive features of the classes analyzing the training set, then the construction of the classifier turns into a trivial procedure.

The process of constructing a classifier can be formally stated as follows (Fig. 2):

$$DW \xrightarrow{P1} TD \xrightarrow{P2} TS \xrightarrow{KD} Ps \xrightarrow{P4} Cl$$

where DW – data warehouse; TD – target dataset; TS – training set; Ps – discovered regularities presented as patterns; Cl – classifier; P1 (Procedure 1) - definition of the classification goal and formation of the target data set; P2 (Procedure 2) – building a training set; KD (Procedure 3) – Knowledge Discovery procedure; P4 (Procedure 4) – classifier construction based on the detected patterns.

It is proposed to implement the process of constructing a classifier based on the interaction of three specialized agents.

The first agent is a Training Set Builder (TSB-agent). As a result of its actions, a *training set* will be formed. The initial data for the TSB-agent are the alphabet of classes, the set of observed features, and an a priori dictionary of features. Based on objects of classes, agent forms a *training set*.

Then control is passed to the Knowledge Discovery-agent (KD-agent). In automatic mode, agent searches for combinations (*ensembles*) of features from a priori dictionary of features, which ensure a distinction between classes.

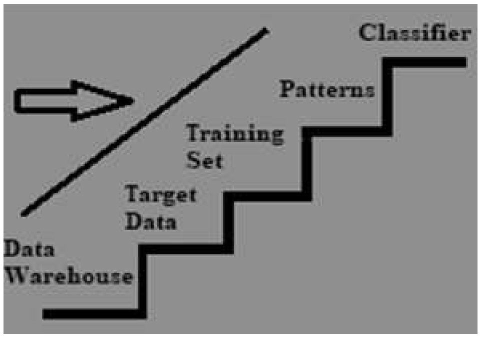


Figure 2. Stages of a classifier construction.

Let's note that if the PDF includes n features, then the number of possible combinations is $2^n - 1$. The KD-agent operating algorithm is detailed in a paper [13].

The final stage of constructing a classifier is performed by the Classifier Builder Agent (CB-agent). It receives a set of combinations of features from the KD-agent and automatically builds class patterns (in the form of cluster structures) and forms a decision rule (on whether the observed object belongs to a certain class).

IV. An Example of Constructing a Classifier

Let's demonstrate the results of agents' interaction on the example of analyzing training set data in order to identify hidden patterns and construct a classifier.

Example. Let the given:

- two classes of five-digit integers **EOOE** and **OOEE**, where the numbers EOOE are such that in tens and hundreds one digit is even and the other is odd, and the numbers OOEE are such that in tens and hundreds both digits are either even or odd;
- a priori dictionary of features F = {units, tens, hundreds, thousands, tens of thousands};
- a training set of five-digit integers that contains 2000 EOOE-numbers and 2000 OOEE-numbers.

Table 1 partially presents the integers from the training set used in the experiment.

Table 2 shows the study results of the intersection of class patterns based on a combination of features **hundreds–tens**, where

$$NE_i = \textit{Number of } EOOE_i$$
$$NO_i = \textit{Number of } OOEE_i$$

$$a_i = \begin{cases} N_i + M_i, N_i = 0 \vee M_i = 0 \\ 0, N_i > 0 \wedge M_i > 0 \end{cases}$$

$$Intersection = \frac{4000 - \sum_{i=1}^{100} a_i}{4000} * 100\%$$

Table I. Training set for the experiment

n/n	Number of EOOE	Number of OOEE
1	14**10**4	79**39**9
2	03**50**5	51**08**8
3	07**34**1	64**82**2
4	41**50**2	72**59**8
5	71**23**4	12**08**3
...	...	...
1999	40**72**4	01**02**7
2000	53**34**7	20**93**4

Table II. Experiment results

hundreds tens	Number EOOE	Number OOEE	hundreds tens	Number EOOE	Number OOEE
0,1	56	0	5,0	37	0
0,3	38	0	5,2	41	0
0,5	39	0	5,4	54	0
0,7	46	0	5,6	43	0
0,9	44	0	5,8	31	0
1,0	31	0	6,1	36	0
1,2	36	0	6,3	50	0
1,4	43	0	6,5	41	0
1,6	47	0	6,7	42	0
1,8	39	0	6,9	36	0
2,1	41	0	7,0	34	0
2,3	39	0	7,2	35	0
2,5	32	0	7,4	36	0
2,7	39	0	7,6	47	0
2,9	44	0	7,8	48	0
3,0	44	0	8,1	22	0
3,2	41	0	8,3	43	0
3,4	33	0	8,5	35	0
3,6	42	0	8,7	34	0
3,8	47	0	8,9	48	0
4,1	38	0	9,0	34	0
4,3	40	0	9,2	34	0
4,5	35	0	9,4	35	0
4,7	51	0	9,6	49	0
4,9	36	0	9,8	34	0
0,0	0	45	5,1	0	0
0,2	0	34	5,3	0	39
0,4	0	48	5,5	0	39
0,6	0	47	5,7	0	40
0,8	0	48	5,9	0	39
1,1	0	32	6,0	0	41
1,3	0	43	6,2	0	51
1,5	0	39	6,4	0	34
1,7	0	46	6,6	0	39
1,9	0	32	6,8	39	0
2,0	0	46	7,1	0	46
2,2	0	46	7,3	0	41
2,4	0	38	7,5	0	48
2,6	0	36	7,7	0	38
2,8	0	34	7,9	0	55
3,1	0	36	8,0	0	36
3,3	0	28	8,2	0	36
3,5	0	46	8,4	0	35
3,7	0	39	8,6	0	44
3,9	0	39	8,8	0	42
4,0	0	39	9,1	0	49
4,2	0	39	9,3	0	38
4,4	0	37	9,5	0	41
4,6	0	36	9,7	0	31
4,8	0	39	9,9	0	32

Table 2 shows that the numbers of the OOEE-class do not have **odd-even** or **even-odd** combinations in hundreds-tens, and the numbers of the EOOE-class do not have **odd-odd** or **even-even** combinations in hundreds-tens, since $Intersection = 0\%$.

As a result, the classifier is constructed on the basis of the following discovered pattern:

IF **((hundreds–tens = odd–tens)** ***or***
(hundreds–tens = tens–odd))
THEN **EOOE** ***ELSE*** **OOEE**

V. Conclusion

The paper presents the process of automatic construction of a classifier based on the interaction of three agents of the knowledge ecosystem.

Based on the class alphabet, a set of observed features, and an a priori dictionary of features, the TSB-agent forms a training set and passes it to the KD-agent. KD-agent automatically discovers a set of combinations of features ensuring distinguishing of classes and delivers it to the CB-agent. The final construction of the classifier by the CB-agent is also performed automatically.

The effectiveness of the proposed method for automatically constructing a classifier is demonstrated on the example of processing model data.

References

[1] E.P. Odum, Osnovy ekologii [Fundamentals of Ecology], Moscow: Mir, 1975, 741 p.

[2] Digital ecosystem [Electronic resource], Available at: https://en.wikipedia.org/wiki/Digital_ecosystem (accessed 2022, May).

[3] P. Kumar, V. Jain, V. Ponnusamy, The Smart Cyber Ecosystem for Sustainable Development, Wiley-Scrivener Publishing, 2021, 480 p.

[4] E. Curry, A. Metzger, S. Zillner, J.-C. Pazzaglia, A.C. Robles The Elements of Big Data Value: Foundations of the Research and Innovation Ecosystem, Springer, 2021, 411 p.

[5] ECM-Journal [Electronic resource], Available at: https://ecm-journal.ru/material/cifrovaja_ehkosistema_modnyjj_termin_ili_novaja_realnost (accessed 2022, May).

[6] S. Szoniecky, Ecosystems Knowledge: Modeling and Analysis Method for Information and Communication, Wiley, 2018, 232 p.

[7] C.W. Choo, N. Bontis, The Strategic Management of Intellectual Capital and Organizational Knowledge, Oxford University Press, 2002, 748 p.

[8] Knowledge ecosystem [Electronic resource], Available at: https://en.wikipedia.org/wiki/Knowledge_ecosystem (accessed 2022, May).

[9] AIportal [Electronic resource], Available at: http://www.aiportal.ru/articles/multiagnt-systems/weak-and-strong-intelligent-agent.html (accessed 2021, October).

[10] M. Wooldridge, An Introduction to MultiAgent Systems, John Wiley & Sons, 2009, 488 p.

[11] V. Tarasov, Artificial Meta-Intelligence: a Key to Enterprise Reengineering // Proc. of the Second Joint Conference on Knowledge Based Software Engineering (JCKBSE'96) (Sozopol, Bulgaria, Sept. 21-22, 1996), Sofia BAIA, 1996, pp.15-24.

[12] Data mining [Electronic resource], Available at: https://en.wikipedia.org/wiki/Data_mining (accessed 2022, May).

[13] V. Krasnoproshin, V. Rodchenko, A. Karkanitsa, "Specialezed KD-agent for knowledge ecosystems" in Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], BSUIR, Minsk, 2021, pp.59–62.

Автоматическое построение классификаторов агентами экосистемы знаний

Краснопрошин В.В., Родченко В.Г., Карканица А.В.

Под цифровой экосистемой понимают распределенную, адаптивную, открытую социотехническую систему, которая обладает свойствами самоорганизации, масштабируемости и устойчивости. Высокий уровень адаптивности и самоорганизация цифровой экосистемы могут быть обеспечены только встроенной по принципу матрешки экосистемой знаний. Главной целью экосистемы знаний является эффективное управление знаниями, которое достигается в результате совершенствования среды взаимодействия участников системы, упрощения процесса принятия решений и стимулирования инноваций.

Базовыми элементами экосистемы знаний являются программные агенты. Они “живут” в среде экосистемы: получают и анализируют данные об окружающих событиях, интерпретируют их и выполняют команды, которые воздействуют на среду.

В работе описан процесс автоматического построения классификаторов на основе взаимодействия агентов экосистемы знаний. Исходной информацией для совместной работы агентов являются алфавит классов, априорный словарь признаков (АСП) и хранилище данных. Эффективность предложенного подхода демонстрируется на примере обработки модельных данных.

Received 14.11.2022

The structure of next-generation intelligent computer system interfaces

Mikhail Sadouski
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: sadovski@bsuir.by

***Abstract*—This article deals with the structure of adaptive multimodal interfaces of next-generation intelligent computer systems, which provide a transition from the paradigm of literate user to the paradigm of equal cooperation between the user and the intelligent system, which will increase the efficiency of human-machine interaction.**

***Keywords*—adaptive intelligent multimodal interface, OSTIS, ostis-system interface, next-generation intelligent computer systems**

I. Introduction

The organisation of user interaction with computer systems (including intelligent computer systems) has a significant impact on user experience, user satisfaction, and the effectiveness of the automation of human activity.

One of the key properties of next-generation intelligent computer systems is their **interoperability** - the ability to interact effectively. Such systems are autonomous and self-sufficient actors on par with humans. However, at the core of modern organization of user interaction with a computer system is the paradigm of **literate user**, who knows how to manage the system and is fully responsible for the quality of interaction with it. The variety of forms and types of interfaces leads to the need for the user to adapt to each particular system and learn the principles of interaction with it in order to solve the required tasks.

The current stage of the field of Artificial Intelligence requires the transition from the paradigm of competent control of the tool used to the paradigm of **equivalent cooperation, partner-like interaction** of intellectual computer system with its user to increase the efficiency of interaction. The friendliness of the user interface should be determined by the ability of the system to adapt to the characteristics and qualifications of the user, its ability to resolve any problems the user might experience during the dialogue with the intelligent computer system, and the way it is concerned with the improvement of the user's communication skills. Consequently, it is necessary to move away from adapting the user to the system (by teaching them how to use it) towards the adaptation of the interface itself to the goals, tasks and characteristics of a particular user in real time. [1]

Thus, the key problems at the current stage are:

- the necessity for the user to learn how to interact with each particular system;
- the lack of partnership between the user and the system (the system is controlled by the user), which leads to the user having to be a constant initiator of interaction;
- tha lack of a system's adaptation to each individual user and the environment in order to maximise the user's comfort while using the system.

In order to solve these problems, this article discusses the principles of organising partner-like interaction between a user and an intelligent system, as well as the principles of building next-generation intelligent computer system interfaces that provide a transition to the paradigm of equal cooperation.

II. State of the art

An interface is a set of technical, software and methodological (protocols, rules, conventions) tools, which enable the exchange of information between the user and devices and programmes, as well as between devices and other devices and programmes. [2]

Broadly speaking, it is a way (standard) of interacting between objects. In technical terms, an interface defines the parameters, procedures and characteristics of interaction between objects.

Interfaces come in many varieties. They differ in the nature of the systems that interact with each other, implementation, and functions.

Regardless of the type of interface, the interaction of the computer system with its environment is facilitated by sensors and effectors.

A sensor (or receptor) of a system is a component of a cybernetic system that generates information in the system's memory about the current value of a property (characteristic, parameter) corresponding to that component of the physical environment of the cybernetic system that is directly adjacent to the said component.

An effector is a component of a cybernetic system that is able to change its state in order to directly affect its physical shell and the external environment.

It is customary to distinguish the following types of interfaces:

- physical interface;
- software interface;
- user interface.

A physical interface is a device that converts signals and transmits them from one piece of equipment to another. A physical interface is defined by a set of electrical connections and signal characteristics.

A software interface is a system of unified connections designed to exchange information between components of a computer system. The software interface defines a set of required procedures, their parameters and how to call them.

A user interface is the combination of hardware and software that enables the exchange of information between a user and a computer system. [3]

This article will focus on the user interface, although many of the principles can be applied to other types of interfaces. A distinction is made between the following types of user interfaces:

- command user interface;
- WIMP interface;
- SILK interface. [4]

A command user interface is a type of interface in which a person gives "commands" to a computer and the computer executes them and prints the result to the person. The command user interface is implemented in the form of batch technology and command line technology.

A WIMP interface (graphical user interface: Window, Image, Menu, Pointer) is an interface in which program functions are represented by graphical screen elements. A characteristic feature of this type of interface is that the dialogue with the user is not with the help of commands but with the help of graphic images - menus, windows, and other elements. Although this interface also gives commands to the machine, this is done "indirectly", through graphic images. This type of interface is implemented on two technological levels: there can be simple graphical interfaces and "pure" WIMP interfaces.

The features of a simple graphical interface are as follows: highlighting screen areas; overriding keyboard keys depending on the context; using manipulators and keyboard keys to control the cursor. A WIMP interface proper is characterised by the following features: all interaction with programs, files and documents takes place in windows; all objects are represented as icons; all actions with objects are performed using menus; extensive use of manipulators to point to objects.

A SILK interface (natural language interface: Speech, Image, Language, Knowlege) is an interface in which the user dialogs with the system in natural language. This type of interface is closest to the usual, human form of communication. The system finds commands for itself by analyzing human speech and finding key phrases in it. The result of the commands is also converted into a human-understandable form.

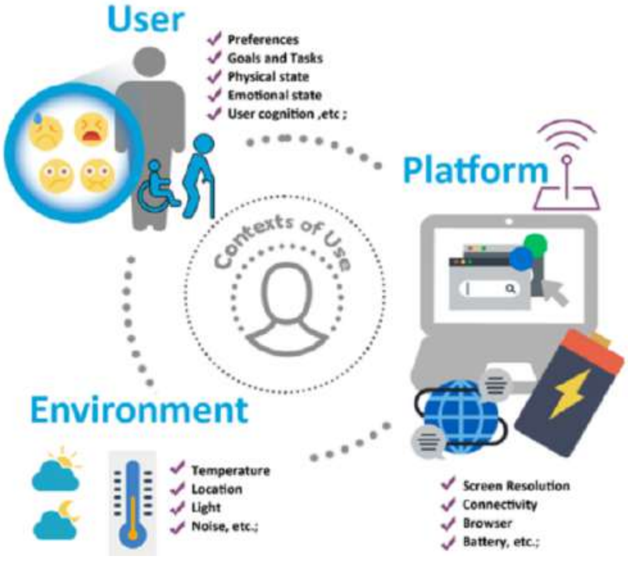


Figure 1. The context-of-use for adaptive UI

Dialogues form the basis of interaction in the user interface. Dialogue in this case is understood as a regulated exchange of information between the user and the system, carried out in real time and aimed at completing a specific task collaboratively. The exchange of information is carried out by transmission of messages.

Tasks to be solved by interfaces (interface tasks) include:

- analysing input information;
- managing effectors.

The quality with which a cybernetic system solves tasks is conditioned by:

- the cybernetic system's ability to understand sensory information;
- the cybernetic system's ability to understand the messages it receives;
- the ability of the cybernetic system to operate independently in the external environment.

The interface of next-generation intelligent computer systems must be able to interact with the user on an equal footing, adapt to the user's characteristics, and accept different types of information input. This kind of interface design is often described as *adaptive*, *intelligent* and *multimodal*.

An adaptive user interface is a set of software and hardware that allows the user to use the system in the most efficient way by automatically adapting the interface to the user's needs and context. [5]

Generally, the context-of-use consists of user, platform, and environment, as shown in Figure 1. [6]

Functionality and parameters of the interface can be adjusted either manually by the user or automatically by the system based on the information about the user. Thus a distinction must be made between adaptive and adaptable systems, terms that are not synonymous, although it is quite common to see these terms used interchangeably in the literature. [7]

In adaptive systems, any adaptation is predefined and can be changed by users before the system starts up. In contrast, in adaptive systems any adaptation is dynamic, i.e. it occurs at the same time as the user interacts with the system, and depends on the user's behaviour. But a system can also be adaptive and adaptive at the same time. [8]

The disadvantage of editing the interface manually is that the user needs to be reasonably familiar with both the system itself and the means to modify its interface.

The term adapted interface can also be found in the literature. Adapted user interfaces [9] are user interfaces adapted to the end-user at design time, with no adaptation changes occurring in run time.

Intelligent User Interface (IUI) - a user interface that can assume what actions the user could perform next and present information based on this assumption. [10]

An intelligent interface should perform the following functions:

- communication function. Communication can take place on the basis of text messages, all kinds of voice input/output systems, graphical interaction tools, etc.
- automatic programme synthesis function. The user message must be converted into a working programme that the computer system can execute.
- justification function. A user who has little or no knowledge of how a computer system converts his task into a working program and what methods it uses to arrive at a solution should be able to know how the system arrived at the resulting solution. He can ask how his task was converted into a program, what method was used to find the solution, how this solution was arrived at, and how it was interpreted in the output. Thus, the justification function includes both an explanation function and a trust function, the purpose of which is to increase the user's trust in the system.
- education function. Next-generation intelligent computer systems must have special means by which the user gradually learns how to use the system and the subtleties of successful communication with it. [11]

As we have seen, the terms "intelligent interface" and "adaptive interface" are different. However, in various articles these concepts are treated as synonyms.

The term intelligent user interface is often used along with various adapt* terms, as reported by a meta-study conducted by Volkel et al. [12], where authors confirmed that the studies might call an entity both "intelligent" and "adaptive". The concurrence can even be observed in use of the term adaptive intelligent user interfaces. Though this term is used infrequently, it describes user interfaces with intelligent adaptive mechanisms capable of monitoring the user behavior and adapting the user interface accordingly, outside of the predefined rules. Many intelligent interfaces can be described as adaptive interfaces, though not all adaptive interfaces are intelligent. IUIs can be associated with intelligent systems, i.e., systems that give appropriate problem-solving responses to problem inputs, even if such inputs are new and unexpected.

An often-made mistake is to confuse an IUI with an intelligent system. A system exhibiting some form of intelligence is not necessarily an intelligent interface. There are many intelligent systems with very simple non-intelligent interfaces and the fact that a system has an intelligent interface does not say anything about the intelligence of the underlying system (Figure 2).

Unfortunately, the boundary between a system and its interface is not always very clear. Often the technology used in an IUI is also part of the underlying system, or the IUI may even form the entire system itself. For example, a speech recognition system can be part of an intelligent interface to a system, but it can also be the complete system depending on how you look at it. If an IUI can be regarded as a system on its own, then it is by definition an intelligent system.

A multimodal interface is a user interface designed to handle two or more combined modes of user input, such as speech, pen, touch, hand gestures, gaze, etc., in a manner coordinated with the output of a multimedia system.

Interaction with most traditional computer systems is done via keyboard and mouse (touchpad, stylus). The user interface of such systems generally does not store information about the user model, the history of the user's actions, and the model of the subject domain. Traditional user interfaces also do not contain an adaptation module. Figure 3 shows the architecture of traditional user interfaces.

The overall architecture of an adaptive intelligent multimodal user interface, in turn, generally looks as shown in Figure 4.

Input coming from the keyboard, mouse, microphone, camera, or possibly some other input device is recorded and then (pre-)processed. Processing includes labeling of events and other interesting input features. After each input modality has been analyzed, the separate modalities are fused together and evaluated. Note that in some cases it is desirable to do the fusion of input streams before the input processing, depending on the application and the features that need to be detected. Once we know what input is coming in, we can start to determine the necessary course of action. First we have to evaluate what to do in the current situation. If there is information

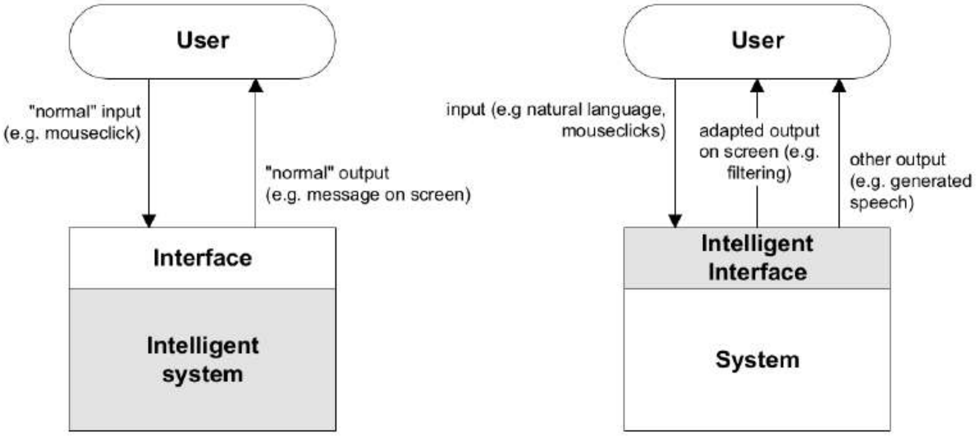


Figure 2. An intelligent system versus an intelligent interface

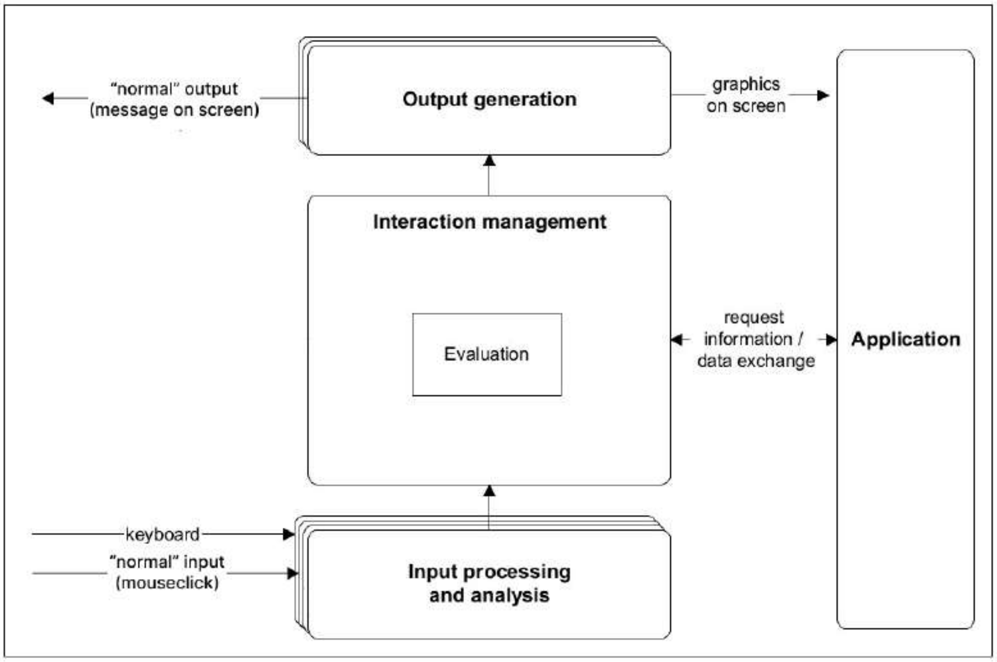


Figure 3. Traditional user interface architecture

missing or if the user has requested information (e.g. the recorded speech contained a question from the user) this information will be requested from the application or some other external source. Usually there is an inference mechanism that draws up conclusions and updates the system's information: the user model, his interaction history, and information about the application domain. Once, all the necessary information is available and updated, the system must decide the best alternative for action. In the figure above we have called this adaptation, since usually some form of adaptation of the interface is chosen. Often, evaluation and adaptation occurs simultaneously using one inference engine for both, making the distinction between the evaluation and adaptation process is not quite clear. The chosen action still has to be generated, which is being done in the output generation part. Most IUIs can be created with or fitted in this architecture, although often not all parts need to be explicitly modeled. [13]

Among modern tools for creating adaptive user interfaces, the following can be highlighted, as shown in Figure 5. [14]

Regardless of the means of creating adaptive intelligent multimodal user interfaces, such systems must effectively store and process knowledge about the user, the interaction with the user, and other relevant information. Most of

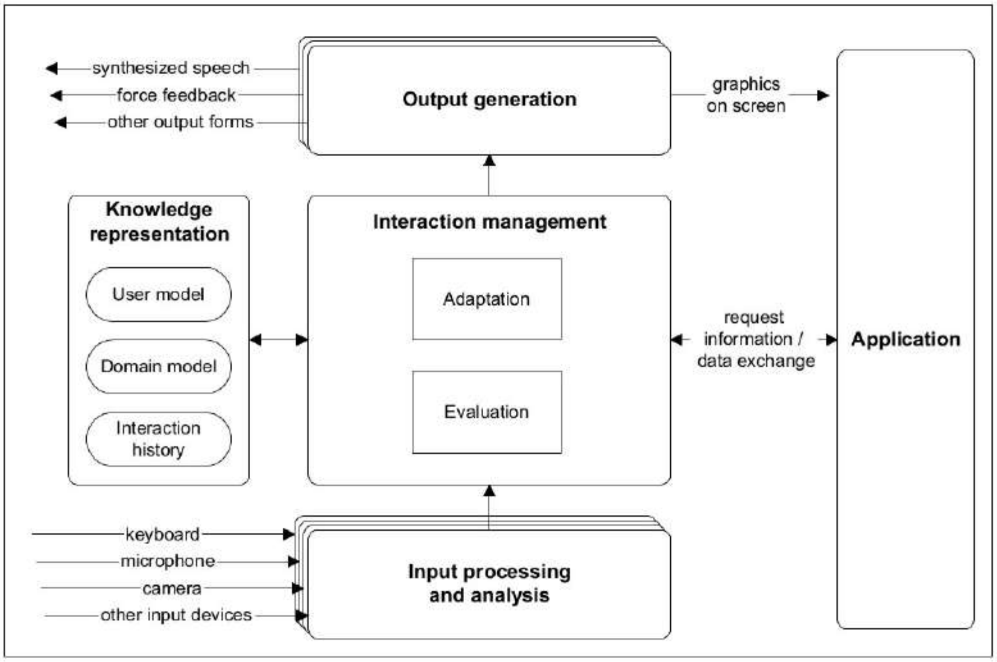
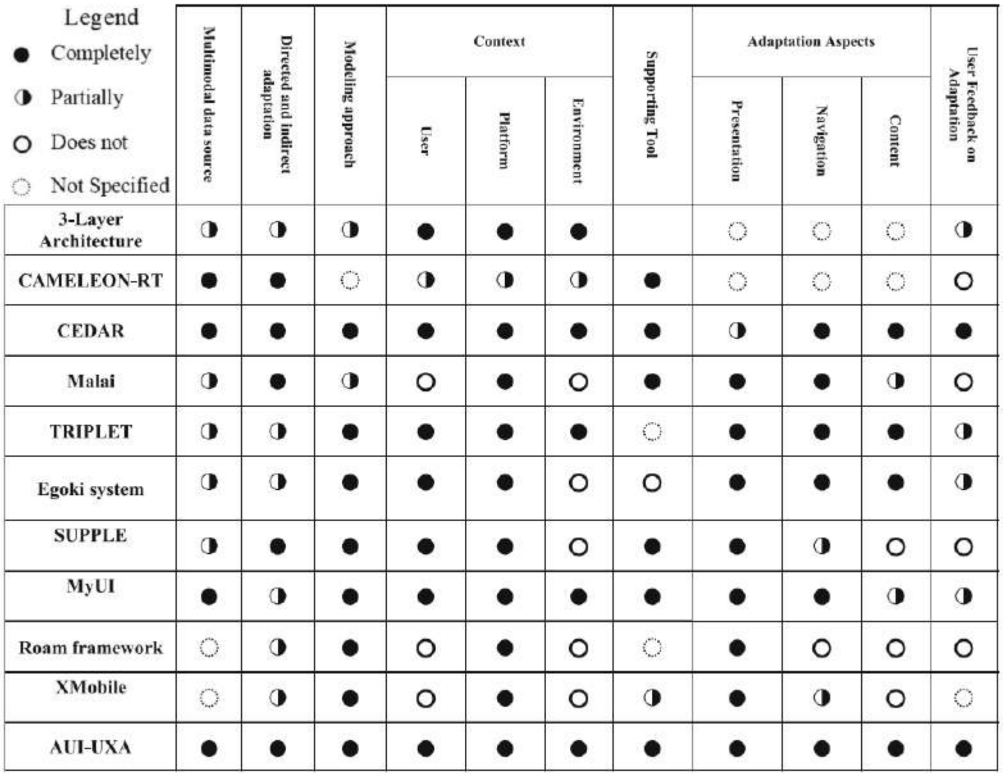


Figure 4. Intelligent user interface architecture



Legend

● Completely

◑ Partially

○ Does not

◌ Not Specified

	Multimodal data source	Directed and indirect adaptation	Modeling approach	Context			Supporting Tool	Adaptation Aspects			User Feedback on Adaptation
				User	Platform	Environment		Presentation	Navigation	Content	
3-Layer Architecture	◑	◑	◑	●	●	●		◌	◌	◌	◑
CAMELEON-RT	●	●	◌	◑	◑	◑	●	◌	◌	◌	○
CEDAR	●	●	●	●	●	●	●	◑	●	●	●
Malai	◑	●	◑	○	●	○	●	●	●	◑	○
TRIPLET	◑	◑	●	●	●	●	◌	●	●	●	◑
Egoki system	◑	◑	●	●	●	○	○	●	●	●	◑
SUPPLE	◑	●	●	●	●	○	●	●	◑	○	○
MyUI	●	◑	●	●	●	●	●	●	●	◑	◑
Roam framework	◌	◑	●	○	●	○	◌	●	○	○	○
XMobile	◌	◑	●	○	●	○	◑	●	◑	○	◌
AUI-UXA	●	●	●	●	●	●	●	●	●	●	●

Figure 5. Existing tools for creating adaptive user interfaces

adaptive UI system use ontological models for storing the information to tailor the UI. It is the ontological approach that allows to:

- create the most complete unified description of the different aspects of the user interface;
- easily integrate various aspects of the user interface;
- make it easier to reuse the interface model.

In ontological approach, it is common to distinguish ontologies and subject domains. The knowledge base of an adaptive intelligent multimodal interface should include at least the following domains:

- Subject domain and ontology of user model;
- Subject domain and ontology of interface components;
- Subject domain and ontology of interface actions;
- Subject domain and ontology of context of use.

Among already existing **user model ontologies**, we can highlight the GUMO ontology [15]. This user model ontology differentiates between:

- physiological state - can change within seconds;
- mental state - can change within minutes;
- emotional state - can change within hours;
- characteristics - can change within months;
- personality - can change within years;
- demographics - can't normally change at all.

H. Paulheim, F. Probst [16] discusses **an interface component ontology**, with the following component types at the top level:

- presentation user interface component;
- decorative user interface component;
- interactive user interface component;
- data-input-component;
- presentation-manipulation-component;
- operation-trigger-component
- container;
- window;
- modal-window;
- non-modal-window.

An ontology can also include a class of component properties that define the appearance of interface elements, ranging from simple properties, such as font, colour, element size, to composite properties, containing sets of interface solutions. [17]

Classification of **interface actions** is presented in [16] and contains the following main classes:

- mouse-action;
- speech-action;
- tangible-action;
- touch-action;
- pen-base-action.

An ontology of context usage is discussed in [18] and describes:

- Users' status:
  - Motion (standing, sitting, walking);
  - Able to listen (yes, no);
  - Able to type (yes, no);
  - Able to talk (yes, no);
  - Able to read (yes, no);
- Natural environment:
  - Lighting (bright, moderately lit, dark);
  - Noise (noisy, quiet)Wind (strong, light, no wind);
  - Weather (sunny, cloudy, rainy);
  - Temperature (hot, warm, cold);
  - Location (in an office, in an airport, on a street, ina library, at home, at a shopping mall);
- Device features:
  - Screen size (big, small);
  - Type of screen (monochrome, colored);
  - Keyboard (large, small, virtual).

It is common to use **intelligent agents** to manage user interaction with the system.

An intelligent agent is one that is capable of flexible autonomous action in order to meet its design objectives. According to this definition, flexible means three things:

- Reactivity: intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives;
- Pro-activeness: intelligent agents are able to exhibit goal directed behavior by taking the initiative in order to satisfy their design objectives;
- Social ability: intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

Intelligent agents are directed towards a single goal, but they possess more knowledge about reasoning within the space of their activity. Knowing when to use other resources (other agents), the preferences of the user or client, constructs for negotiation deals, and other abilities are the marks of an intelligent agent.

The following conclusions can be made, based on our analysis:

- To move to a paradigm of equal cooperation between user and system, interfaces need to be adaptive, intelligent, and multimodal. Existing solutions allow such interfaces to be designed but have a number of shortcomings, which will be presented below.
- The structure of intelligent interfaces includes a knowledge base, a module for managing user interaction with the system.
- Ontological approach is actively used in the design of knowledge bases and some ontologies that are used in the design of intelligent interfaces have already been implemented.
- The module for managing user interaction with the system is usually implemented based on a multi-agent approach.

Disadvantages of existing solutions include:

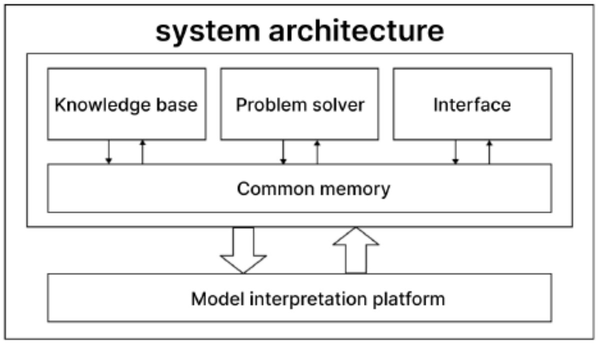


Figure 6. Intelligent system/intelligent interface architecture

- Existing solutions generally involve a question-and-answer principle of interaction.
- Still relevant is the problem of compatibility between the intelligent interface and the intelligent system for which it is being created, due to different tools and methods being used in design and implementation.
- The compatibility of the intelligent interface components (knowledge base and interaction management module) with each other remains a relevant problem.

III. Proposed approach

To address the shortcomings of existing solutions, it is proposed to use an ontological approach based on a semantic model in the design and implementation of an adaptive intelligent multimodal user interface. We propose to view such an interface as a specialized subsystem for solving user interface problems that consists of a knowledge base and an interface problem solver. The model of knowledge base and solver can be described on the basis of a universal unified knowledge representation language, which will ensure compatibility between these components.

An intelligent system for which an intelligent interface is to be created should have a model described using the same unified language as the intelligent interface itself. This will ensure that the intelligent system and its intelligent interface are compatible.

An intelligent interface problem solver should be based on a multi-agent approach, and the agents themselves should be able to initiate actions and messages to the user and other agents.

The architecture of such an intelligent system and an intelligent interface based on the same principles would look as follows (Figure 6).

Thus, we can formulate a list of requirements that the technology necessary to implement this approach should satisfy:

- the technology should support component approach to creating semantic models;
- the technology should allow straightforward integration of various semantic models within a unified system;
- the technology should make it possible to describe different semantic models and various types of knowledge therein using a single format.

As compared to other existing system design technologies, the *OSTIS Technology* meets all the specified requirements. Another advantage of the technology that can be highlighted is that it includes a basic set of ontologies that can serve as the ground for the IUI model being developed.

Thus, within this approach, we propose base the implementation of a framework for building UIs on the *OSTIS Technology*. This technology provides a universal language for the semantic representation (encoding) of information in the memory of intelligent computer systems, called *SC-code*. Texts written in *SC-code* (sc-texts) are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-connectors* and *sc-nodes*, which, in turn, can be subdivided into *sc-edges* or *sc-arcs*, depending on the directivity). The *SC-code alphabet* consists of five main elements that can be used to create SC-code constructions of any complexity as well as to introduce more specific types of sc-elements (for example, new concepts). The memory that stores SC-code constructions is called semantic memory or *sc-memory*. [19].

The architecture of each ostis-system includes a platform for interpreting semantic models of ostis-systems as well as a semantic model of the ostis-system described using SC-code (sc-model of the ostis-system). In turn, the sc-model of the ostis-system includes sc-model of the knowledge base, sc-model of the interface, and sc-model of the problem solver. The principles of the design and structure of knowledge bases and problem solvers are discussed in more detail in [20] and [21], respectively. This article describes the sc-model of the UI, which is included in the sc-model of the interface. Its principles were described in the article [22], which this paper builds upon.

The architecture of the ostis-system is shown in Figure 7.

The proposed architecture for an adaptive intelligent multimodal user interface is shown in Figure 8.

The subject domains of user, context of use, user interface actions, and interface components is proposed to be formalised in the same way as the ontologies discussed in section 2.

The subject domain of user interface is a formalised typology of user interfaces. An example of a fragment of this domain in a user interface knowledge base would look as follows.

user interface

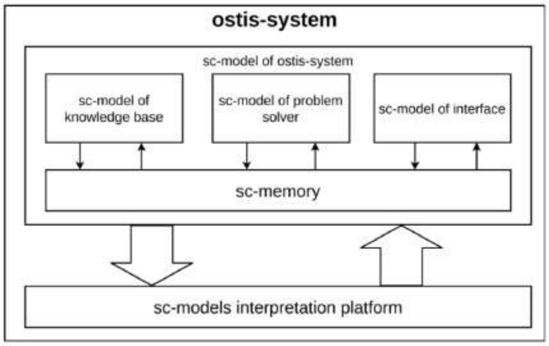

ostis-system
sc-model of ostis-system
sc-model of knowledge base
sc-model of problem solver
sc-model of interface
sc-memory
sc-models interpretation platform

Figure 7. The architecture of the ostis-system

⊃ *graphical user interface*
  ⊃ *WIMP interface*
    ⊃ *ostis-system user interface*
⊃ *command-line interface*
⊃ *SILK interface*
  ⊃ *natural-language interface*
    ⊃ *speech interface*

Within the subject domain of interface design methodologies, it is proposed to formalise the various existing interface design methods, such as:

- designing user interfaces based on ontologies (ontology-driven user interface design);
- ergonomic design methodology;
- goal-oriented design methodology.

Within the interface design tools subject area, it is proposed to formalise existing interface design tools such as:

- tools to support the creation of an interface by writing code;
- interactive tools;
- tools based on creating an interface by linking separately created components.

[23]

The subject domain of messages is a formalised typology of messages such as declarative, interrogative, etc.

Within the subject domain of logical rules for interface adaptation, it is proposed to formalise a typology of logical rules on the basis of which interface adaptation to the user will take place.

The subject domain of internal interface agent actions describes the classification of possible actions in the ostis system [3]. A fragment of the knowledge base containing this domain is given below.

action in sc-memory
⊃ *action of interpreting a program stored in sc-memory*
⊃ *action of setting the mode of the ostis-system*
⊃ *action in sc-memory initiated by a question*
⊃ *action of editing a file stored in sc-memory*
⊃ *action of editing the ostis-system knowledge base*

In [24], a problem solver model has been described. The problem solver model should also include a user interface adaptation and evaluation module.

Any ostis-system can integrate an intelligent interface according to the proposed architecture. But it is also important to clarify the concept of user interface in the context of the OSTIS Ecosystem.

The OSTIS Ecosystem is a socio-technical network of interactions between:

- ostis-systems themselves;
- users of the specified ostis-systems (both end-users and developers);
- some computer systems that are not ostis-systems (they can be used as additional information resources or services).

The objectives of the OSTIS Ecosystem are:

- rapid implementation of all agreed upon changes in ostis-system;
- permanent maintenance of a high-level mutual understanding between all the systems that make up the OSTIS Ecosystem and all their users;
- corporate solution of various complex tasks requiring the coordination of several (most often a priori unknown) ostis-systems, and possibly some users.

The OSTIS Ecosystem has a concept of a personal ostis-assistant, which is an ostis-system that is a personal assistant to a member of OSTIS Ecosystem, i.e. an ostis-system that mediates the person's interactions with the members of all the collectives (ostis-communities) of which the person is himself a member.

Since user interaction with the OSTIS Ecosystem takes place only via a personal assistant, an adaptive intelligent multimodal user interface is required only for ostis-systems that are personal assistants but not for all ostis-systems.

A model of the user, their activities, etc. in this context should only be stored within the user's personal assistant memory and shared with other systems as needed.

IV. Conclusion

The article discusses the principles of organising partne-like interaction between a user and an intelligent system and the principles of constructing interfaces for next-generation intelligent computer systems that provide a transition to the paradigm of equal cooperation between a user and a user interface.

The following conclusions have been drawn as a result of the analysis:

- In order to move to a paradigm of equal cooperation between a user and a system, interfaces need to be adaptive, intelligent, and multimodal. Existing solutions allow such interfaces to be designed but have a number of shortcomings, which will be presented below.

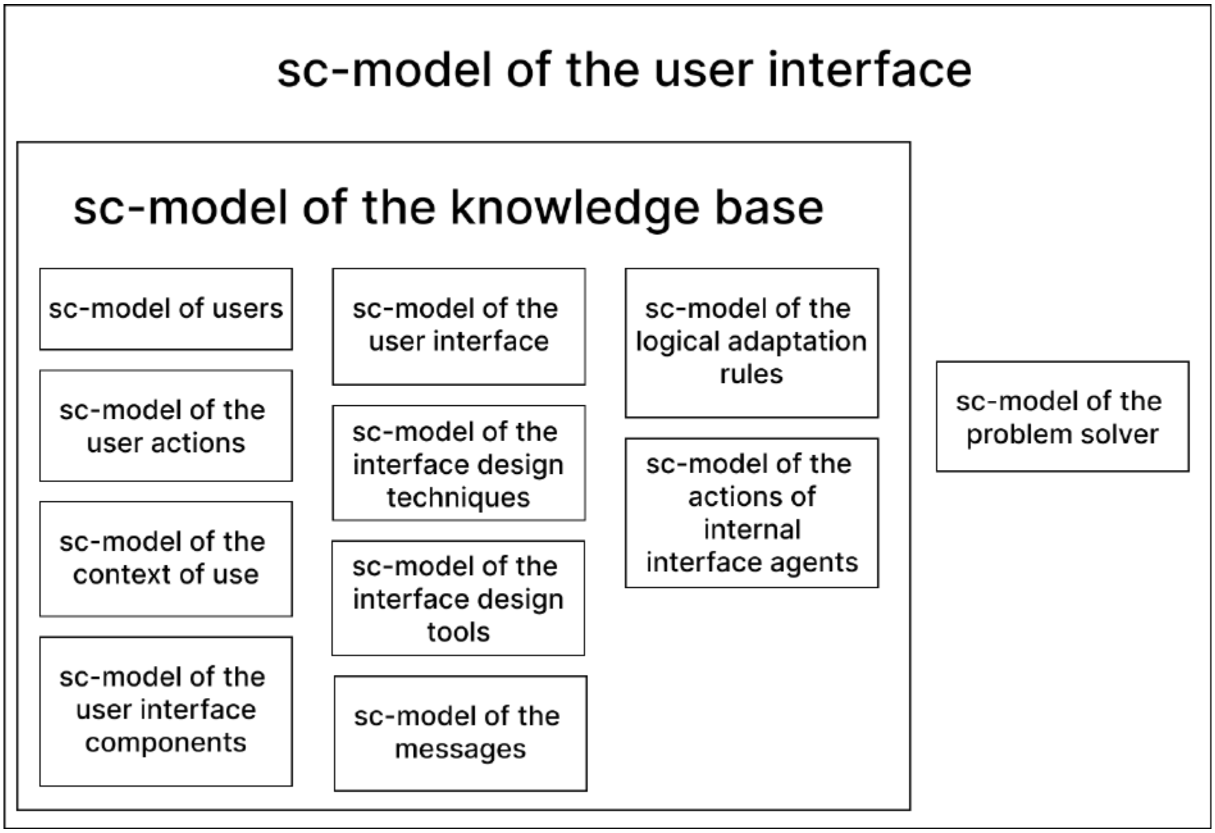

sc-model of the user interface
sc-model of the knowledge base
sc-model of users
sc-model of the user actions
sc-model of the context of use
sc-model of the user interface components
sc-model of the user interface
sc-model of the interface design techniques
sc-model of the interface design tools
sc-model of the messages
sc-model of the logical adaptation rules
sc-model of the actions of internal interface agents
sc-model of the problem solver

Figure 8. Intelligent interface architecture

- The structure of intelligent interfaces includes a knowledge base and a module for managing user interaction with the system.
- Ontological approach is actively used in the design of knowledge bases, and some ontologies have already been implemented and utilized in the design of intelligent interfaces.
- The module for managing user interaction with the system is usually implemented based on a multi-agent approach.

The following shortcomings of existing solutions have been highlighted:

- Existing solutions generally involve a question-and-answer principle of interaction.
- Still relevant is the problem of compatibility between the intelligent interface and the intelligent system for which it is being created, due to different tools and methods being used in design and implementation.
- The compatibility of the intelligent interface components (knowledge base and interaction management module) with each other remains a relevant issue.

We proposed an ontological approach based on a semantic model that can be used in the design and implementation of an adaptive intelligent multimodal user interface. The approach is based on the OSTIS Technology, which will provide:

- compatibility of an intelligent interface with an intelligent system;
- compatibility of an intelligent interface components with each other;
- user interaction with the system through an intelligent interface on the principle of equality.

The architecture of an intelligent interface was proposed, its components and its application in the context of the OSTIS Ecosystem have been discussed in detail.

V. Acknowledgment

The author would like to express his gratitude to the scientific teams of the Departments of Intellectual Information Technologies of the Belarusian State University of Informatics and Radioelectronics, and Brest State Technical University. This work was partially supported by BRFFR (BRFFR-RFFR No.F21RM-139).

References

[1] T. A. Fomina and G. M. Novikova, “Proektirovanie adaptivnogo interfejsa is dlya podderzhki deyatel’nosti obrazovatel’nogo uchrezhdeniya,” *Vestnik Altajskoj akademii ekonomiki i prava*, vol. 6, no. 1, pp. 125–133, 2020. [Online]. Available: https://vaael.ru/ru/article/view?id=1174

[2] A. Starostin, *Tehnicheskie sredstva avtomatizacii i upravleniya*. Yekaterinburg: Ural University Press, 2015.

[3] V. V. Golenkov, N. A. Gulyakina, D. V. Shunkevich, *Open technology for ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, G. V.V., Ed. Minsk: Bestprint, 2021.

[4] T. Brusencova and T. Kishkurno, *Proektirovanie interfeisov polzovatelya : posobie dlya stud. vuzov*. Minsk: BSTU, 2019.

[5] I. M. Ismagilova and S. Valeev, "Postroenie dinamicheskih adaptivnih interfeisov informacionno-upravlyayuschih sistem na osnove metodov iskusstvennogo intellekta," *Vestnik Ufimskogo gosudarstvennogo aviacionnogo tehnicheskogo universiteta*, vol. 9, pp. 122–130, May 2018.

[6] J. Hussain, A. U. Hassan, H. Bilal, R. Ali, M. Afzal, S. Hussain, J. Bang, O. Banos, and S. Lee, "Model-based adaptive user interface based on context and user experience evaluation," *Journal on Multimodal User Interfaces*, vol. 12, p. 17, 02 2018.

[7] I. M. Ismagilova and S. Valeev, "Postroenie adaptivnih interfeisov v slojnih raspredelennih tehnicheskih sistemah s primeneniem statisticheskih metodov," *Vestnik Ufimskogo gosudarstvennogo aviacionnogo tehnicheskogo universiteta*, vol. 9, pp. 122–130, May 2018.

[8] M. Montero and E. Gaudioso, *Adaptable and Adaptive Web-Based Educational Systems : Encyclopedia of human computer interaction*. UK: Liverpool John Moores University, 2005.

[9] E. Schlungbaum, "Individual user interfaces and model-based user interface software tools," in *IUI '97*, 1997.

[10] S. Brdnik, T. Heričko, and B. Šumak, "Intelligent user interfaces and their evaluation: A systematic mapping study," *Sensors*, vol. 22, no. 15, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/15/5830

[11] (2022, Sep) Intellektual'nye interfejsy dlya evm novyh pokolenij. [Online]. Available: https://alllink.ru/xt3m-327/8141776/threads.w1yuv.php

[12] S. T. Völkel, C. Schneegass, M. Eiband, and D. Buschek, "What is "intelligent" in intelligent user interfaces? a meta-analysis of 25 years of iui," ser. IUI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 477–487. [Online]. Available: https://doi.org/10.1145/3377325.3377500

[13] P. Ehlert, *Intelligent User Interfaces: Introduction and Survey*, 02 2003.

[14] J. Hussain, A. U. Hassan, H. Bilal, R. Ali, M. Afzal, S. Hussain, J. Bang, O. Banos, and S. Lee, "Model-based adaptive user interface based on context and user experience evaluation," *Journal on Multimodal User Interfaces*, vol. 12, 02 2018.

[15] D. Heckmann, E. Schwarzkopf, J. Mori, D. Dengler, and A. Kröner, "The user model and context ontology gumo revisited for future web 2.0 extensions," vol. 298, 01 2007.

[16] H. Paulheim and F. Probst, *UI2Ont—A Formal Ontology on User Interfaces and Interactions*, 01 2013, pp. 1–24.

[17] V. Gribova and V. Strekalev, "Ontologies for development and generation adaptive user interfaces for knowledge base editors," *Ontology of Designing*, vol. 12, pp. 200–217, 02 2022.

[18] J. Kong, W. Zhang, N. Yu, and X. Xia, "Design of human-centric adaptive multimodal interfaces," *Int. J. Hum.-Comput. Stud.*, vol. 69, pp. 854–869, 12 2011.

[19] V. Golenkov, N. Gulyakina, I. Davydenko, and D. Shunkevich, "Semanticheskie tekhnologii proektirovaniya intellektual'nyh sistem i semanticheskie associativnye komp'yutery [Semantic technologies of intelligent systems design and semantic associative computers]," *Otkrytye semanticheskie tehnologii proektirovanija intellektual'nyh sistem [Open semantic technologies for intelligent systems]*, pp. 42–50, 2019.

[20] I. Davydenko, "Semantic models, method and tools of knowledge bases coordinated development based on reusable components," in *Otkrytye semanticheskie tehnologii proektirovanija intellektual'nyh sistem [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed., BSUIR. Minsk , BSUIR, 2018, pp. 99–118.

[21] D. Shunkevich, "Agentno-orientirovannye reshateli zadach intellektual'nyh sistem [Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems]," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2018, pp. 119–132.

[22] A. Boriskin, M. Sadouski, and D. Koronchik, "Ontology-based design of intelligent systems user interface," *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, vol. 12, pp. 95–106, 02 2017.

[23] B. A. Myers, "A brief history of human computer interaction technology," vol. 5, pp. 44–54, 03 1998.

[24] M. Sadouski, "Semantic-based design of an adaptive user interface," in *Open Semantic Technologies for Intelligent Systems*, V. Golenkov, V. Krasnoproshin, V. Golovko, and D. Shunkevich, Eds. Cham: Springer International Publishing, 2022, pp. 165–191.

Структура интерфейсов интеллектуальных компьютерных систем нового поколения

Садовский М. Е.

В работе рассматривается структура адаптивных мультимодальных интерфейсов интеллектуальных компьютерных систем нового поколения, обеспечивающих переход от парадигмы грамотного пользователя к парадигме равноправного сотрудничества пользователя с интеллектуальной системой, что позволит повысить эффективность человеко-машинного взаимодействия.

Received 10.10.2022

Natural language interfaces of next-generation intelligent computer systems

Artem Goylo
Minsk State
Linguistic University
Minsk, Belarus
Email: artemgoylo@gmail.com

Sergei Nikiforov
Belarusian State University
of Informatics and Radioelectronics
Minsk, Belarus
Email: nikiforov.sergei.al@gmail.com

***Abstract*—The article describes an approach to the implementation of natural language interfaces of next-generation intelligent computer systems built using OSTIS technology, and also proposes a dialogue context model. In this approach, all stages of analysis, including lexical, syntactic and semantic analysis, can be performed directly in the knowledge base of such a system. This approach will effectively solve such problems as managing the global and local contexts of dialogue, as well as resolving linguistic phenomena such as anaphora, homonymy and tackling elliptical phrases.**

***Keywords*—Natural Language Processing, Natural Language Understanding, ontology, context, semantic network, Open Semantic Technology for Intelligent Systems (OSTIS), SC-code (Semantic Computer Code), constituency grammar**

I. Introduction

Currently, there is a large number of different interfaces of computer systems, which complicates interoperability between such systems and people as they need to familiarize themselves with the interface of each new system, which is rarely intuitive.

One of the main features of next-generation intelligent computer systems should be a user interface that can provide effective user interaction with the system, considering that users are often not professionally trained.

Speech is one of the most natural and convenient forms of information transfer between people, which leads to the increasing spread of natural language interfaces [1]. At the present time, no one doubts that this form of human-machine interaction plays and will continue to play a significant role in interaction with various computer systems.

However, it should be noted that a great diversity of languages (both natural and artificial) leads to the need to simplify the process of creating such interfaces for each individual language.

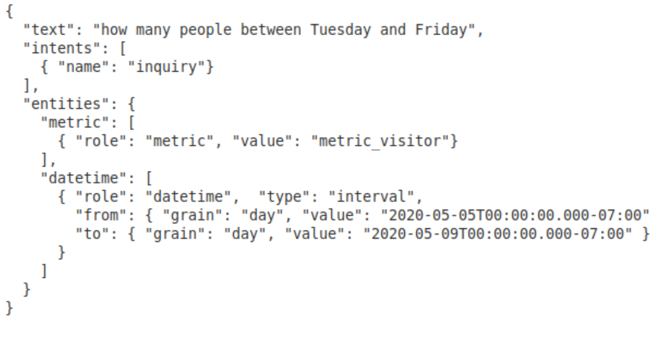
II. State of the art

Most approaches to natural language processing and understanding are based on machine learning [2], [3]. Undoubtedly, for most widely used languages, natural language processing models work very well and are improving every day, but despite the success in this area, this approach has several disadvantages:

- problems when working with different domains, for example, the meanings of words or sentences can be different depending on the subject domain. Thus, models for NLP may work well for a particular subject domain, but not be suitable for general application [4];
- creating a new model requires a large amount of data, and the quality of such data directly affects the quality of the resulting model, which leads to high costs for its training [5] [6];
- the model data is a "black box" because such models do not have the means to provide a description of its inference;
- every such model solves only a small amount of problems, there is no general approach to natural language processing. [4]

These shortcomings of the methods used cause some of the shortcomings of modern systems that implement a natural language interface. For example, despite the fact that now there is a large number of speech assistants created by different companies [7], [8], [9], [10], they have similar drawbacks, namely, an exclusively distributed implementation, due to end-user device performance being insufficient to run resource-intensive models. This in turn leads to privacy issues [11].

The speech understanding submodule of these systems generates a construction that reflects the meaning of the message using a frame model. A simplified example of such a construction is shown in figure 1.



```
{
  "text": "how many people between Tuesday and Friday",
  "intents": [
    { "name": "inquiry"}
  ],
  "entities": {
    "metric": [
      { "role": "metric", "value": "metric_visitor"}
    ],
    "datetime": [
      { "role": "datetime",  "type": "interval",
        "from": { "grain": "day", "value": "2020-05-05T00:00:00.000-07:00" },
        "to": { "grain": "day", "value": "2020-05-09T00:00:00.000-07:00" }
      }
    ]
  }
}
```

Figure 1. Message meaning formalization example

At the same time, other formats are used to present the results of intermediate stages of processing, the modules that implement them do not have any single foundation and interact through specialized software interfaces between them, which leads to incompatibility of the methods for presenting results at various stages of processing and the final result of text processing. This incompatibility, in turn, leads to significant overhead costs in the development of such a system and, in particular, in its modification.

As a solution to the compatibility problem, it is proposed to use an approach to natural language processing based on its formal model in the form of a set of ontologies formed using universal knowledge representation tools. This will contribute to interoperability of the component of natural language processing as a whole with other components of the system, and between parts of the component itself.

The aim of the article is to design an interface model based on an approach to natural language processing that uses ontologies containing a formal description of natural language.

III. Suggested approach

In the suggested approach to the implementation of natural language interfaces, it is proposed to carry out all stages of analysis, including lexical, syntactic and semantic analysis directly in the knowledge base of an intelligent system, presenting the results in a single unified form.

The description of the results of lexical, syntactic and semantic analysis is supposed to be carried out on the basis of the concepts introduced in the following subject domains:

- *Subject domain of the lexicon of natural languages*;
- *Subject domain of syntax of natural languages*;
- *Subject domain of denotational semantics of natural languages*.

However, the specification of these subject domains is not the aim of this article.

We also suggest to introduce a set of concepts to describe the context of the dialogue at various levels. The presence of such contexts will allow storing and using not only the history of the dialog (including the meaning of messages), but also other knowledge that can be used in the course of the dialog, including heterogeneous information about the interlocutor.

We propose to use the representation of knowledge about different languages (including knowledge about their syntax and semantics) in a unified form. This will significantly reduce overhead costs in the development of various systems that use the created ontologies.

In this arcticle it is proposed to base natural language interfaces on *OSTIS Technology* [12]. This technology allows to ensure the compatibility of heterogeneous problem solving models, and reduce the costs of development and modification (including adding a new problem solving model to the system).

Systems developed on the basis of the OSTIS Technology are called ostis-systems. The OSTIS Technology is based on a universal way of semantic representation of information in the memory of intelligent computer systems, called *SC-code*. SC-code texts are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, depending on their directivity, can be *sc-arcs* or textitsc-edges). *SC-code alphabet* consists of five main elements, on the basis of which SC-code constructions of any complexity are built, including the introduction of more specific types of sc-elements (for example, new concepts). The memory that stores SC-code constructions is called semantic memory or *sc-memory*.

Fragments (substructures) of the subject domains and ontologies under consideration, as well as structures related to the knowledge base and problem solver models, will be further shown in the form of SC-code texts (sc-texts).

A *problem solver* of any ostis system (more precisely, the sc-model of the problem solver of an ostis-system) is a hierarchical system of knowledge processing agents in semantic memory (*sc-agents*) that interact with each other exclusively by specifying their actions in the memory [13].

A system of sc-agents over a shared memory is a collective of sc-agents the initiation condition of which is the appearance of a certain construction in the system's memory. In this case, operations interact with each other through the system memory by generating constructions that serve as initiation conditions for another operation.

With this approach, it becomes possible to ensure the flexibility and extensibility of the system functionality by adding or removing a certain set of agents from its composition, without making changes that affect other agents.

A. Subject domain and ontology of natural language interfaces of ostis-systems

Natural language interface – SILK-interface (Speech, Image, Language, Knowledge) – is an interface where the exchange of information between the computer system and the user occurs through dialogue. The dialogue is conducted in one of the natural languages.

natural language interface
$\supset$ *speech interface*

Speech Interface is a SILK interface where information is exchanged through dialogue, during which the computer system and the user communicate using speech. This type of interface is closest to natural communication between people.

In the suggested approach, the following stages of natural language processing can be distinguished:

- lexical analysis;
- syntactic analysis;
- message comprehension.

In turn, lexical analysis includes decomposition of the text into tokens and their mapping to lexemes.

Understanding the message comes down to generating message meaning variants and choosing the correct one based on the context, as well as merging it with this context.

Provided below is the structure of the natural language interface problem solver.

Natural language interface problem solver
$\Rightarrow$ *abstract sc-agent decomposition**:

{• *Abstract sc-agent of lexical analysis*
    ⇒ *abstract sc-agent decomposition**:
        {• *Abstract sc-agent for decomposing text into tokens*
        • *Abstract sc-agent for mapping tokens to lexemes*
        }
• *Abstract sc-agent of syntactic analysis*
• *Abstract sc-agent of message understanding*
}

In turn, *abstract sc-agent of message understanding* is decomposed into:

Message understanding agent
⇒ *abstract sc-agent decomposition**:
    {• *Abstract sc-agent for generating message meaning variants*
    • *Abstract sc-agent for context selection and update*
        ⇒ *abstract sc-agent decomposition**:
            {• *Abstract sc-agent of context resolution*
            • *Abstract sc-agent for choosing the meaning of a message based on the context*
            • *Abstract sc-agent for embedding a message into a context*
            }
    }

The knowledge base must contain a specification of each agent, an example of a fragment of such a specification is shown in figure 2.

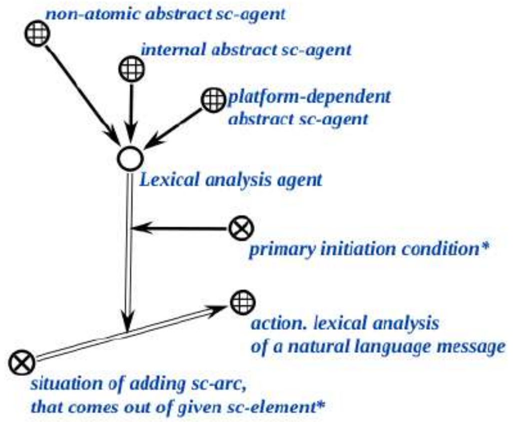

non-atomic abstract sc-agent
internal abstract sc-agent
platform-dependent abstract sc-agent
Lexical analysis agent
primary initiation condition*
action. lexical analysis of a natural language message
situation of adding sc-arc, that comes out of given sc-element*

Figure 2. Agent specification example

B. Subject domain and ontology of lexical analysis of natural language messages included in an ostis-system

action. lexical analysis of a natural language message
⇒ *generalized decomposition**:
    {• *action. decomposition of text into tokens*
    • *action. mapping tokens to lexemes*
    }

From the point of view of an ostis-system, any natural language text is a *file* (i.e. an SC-node with content).

The stage of lexical analysis is the decomposition of the text into a sequence of tokens and the mapping of lexemes to the tokens resulting from this decomposition. It should be noted that these tokens, if necessary, can be compared not with lexemes, but with their subsets included in its morphological paradigm that correspond to certain grammatical categories: case, number, gender, etc.

A result of lexical analysis is shown in figure 3.

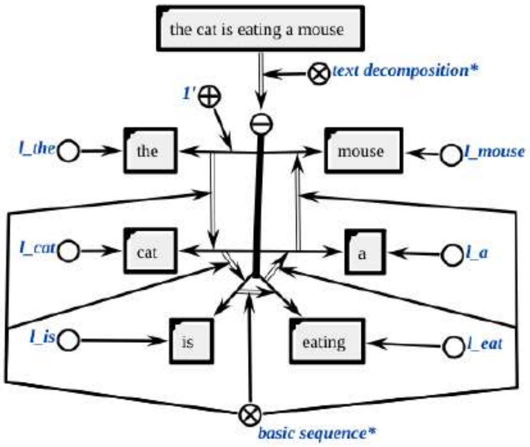


Figure 3. Lexical analysis result example.

To perform lexical analysis, the knowledge base of the system must also contain a lexicon with lexemes and their various word forms.

A lexeme is a unit of the lexicon of a language, a set of all forms of a certain word. An example of a lexeme specification in the knowledge base is shown in figure 4.

C. Subject domain and ontology of syntactic analysis of natural language messages included in an ostis-system

The agent of syntactic analysis performs the translation of the tokenized text into its syntactic structure. At the same time, due to the impossibility of resolving structural ambiguity at the stage of syntactic analysis, its result of syntactic analysis will generally be a set of potential syntactic structures.

An example of a syntactic structure is shown in figure 5.

D. Subject domain and ontology of understanding natural language messages included in an ostis-system

действие. понимание естественно-языкового сообщения

action. natural language message understanding

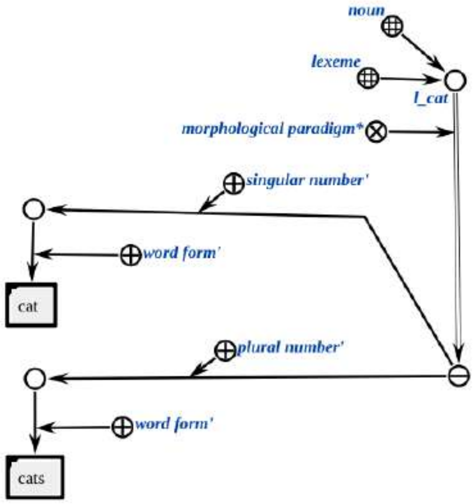


Figure 4. Example of a lexeme in the knowledge base.

⇒ *generalized decomposition**:
{• *action. generation of message meaning variants*
• *action. context selection and update*
⇒ *generalized decomposition**:
{• *action. context resolution*
• *action. selecting the meaning of the message based on the context*
• *action. embedding the message in context*
}
}

Action. message meaning variants deneration is an action during which the formation of a strict disjunction of potentially equivalent structures is carried out.

*Potentially equivalent structure** is a binary oriented relation that connects a structure and a set of structures that could potentially be equivalent to it, however, additional steps are required to reliably determine this.

At the same time, the transition from the result of syntactic analysis to structures potentially equivalent to the message is carried out according to the rules contained in the subject domain of denotational semantics. An example of one of the rules is shown in figure 6.

As a result of this action, a structure is formed in the knowledge base that describes possible variants of the meaning of the message, an example of such a structure based on constituency grammar [14] is given in figure 7. The presence of several such structures is explained by the fact that, in general, several variants of the syntactic structure are generated at the stage of syntactic analysis. The choice of the correct message meaning will be made in the course of the subsequent steps.

It should be noted that, if necessary, the meaning of the message can be generated not only on the basis of its syntactic structure based on constituency grammar, but also on other knowledge about this message, for example, subject-relation-object triples extracted from the text of this message, the result of message classification, etc.

Further steps in the message understanding process are performed based on the context.

Context is as sc-structure containing the knowledge used by the system during one or more dialogs. Generally, this knowledge includes both previously provided in the knowledge base and obtained in the course of operating sensors and / or communicating during a dialog.

dialog context
⊂ *context*
⇒ *subdividing**:
Typology of dialog contexts by scope^
= {• *thematic context*
• *user context*
• *global context*
}

Thematic context is a dialog context containing topic-specific information (information obtained during the dialog on a certain topic, for example, when talking about a certain set of entities).

*A set of thematic contexts of a dialog** is a binary oriented relation, a dialog with the oriented set of its thematic contexts.

User context is a dialog context that contains user-specific information that can be used in a dialog with them on any topic. In general, user context intersects with *the approved part of the KB* (reliable information about the user previously provided in the KB that has passed the necessary moderation), but is not included in it entirely (the part received during the dialog that we are not sure about). An example of intersection of different types of contexts with an approved part of the knowledge base is shown in figure 8.

Global context is a dialog context that contains information that may be needed when conducting a dialog with any user. Global context is a subset of the approved part of the knowledge base that contains the information that may be used in the dialogue. For example, in a dialog with a specific user, it is unnecessary to use:

- proprietary information located in the knowledge base, which is necessary for the system to work but not intended for use in the dialog;
- parts of user contexts of other users.

dialog context
⇒ *subdividing**:
Context typology by knowledge validity period^
= {• *dialog context that does not change during system operation*
• *dialog context that changes during system operation*
}

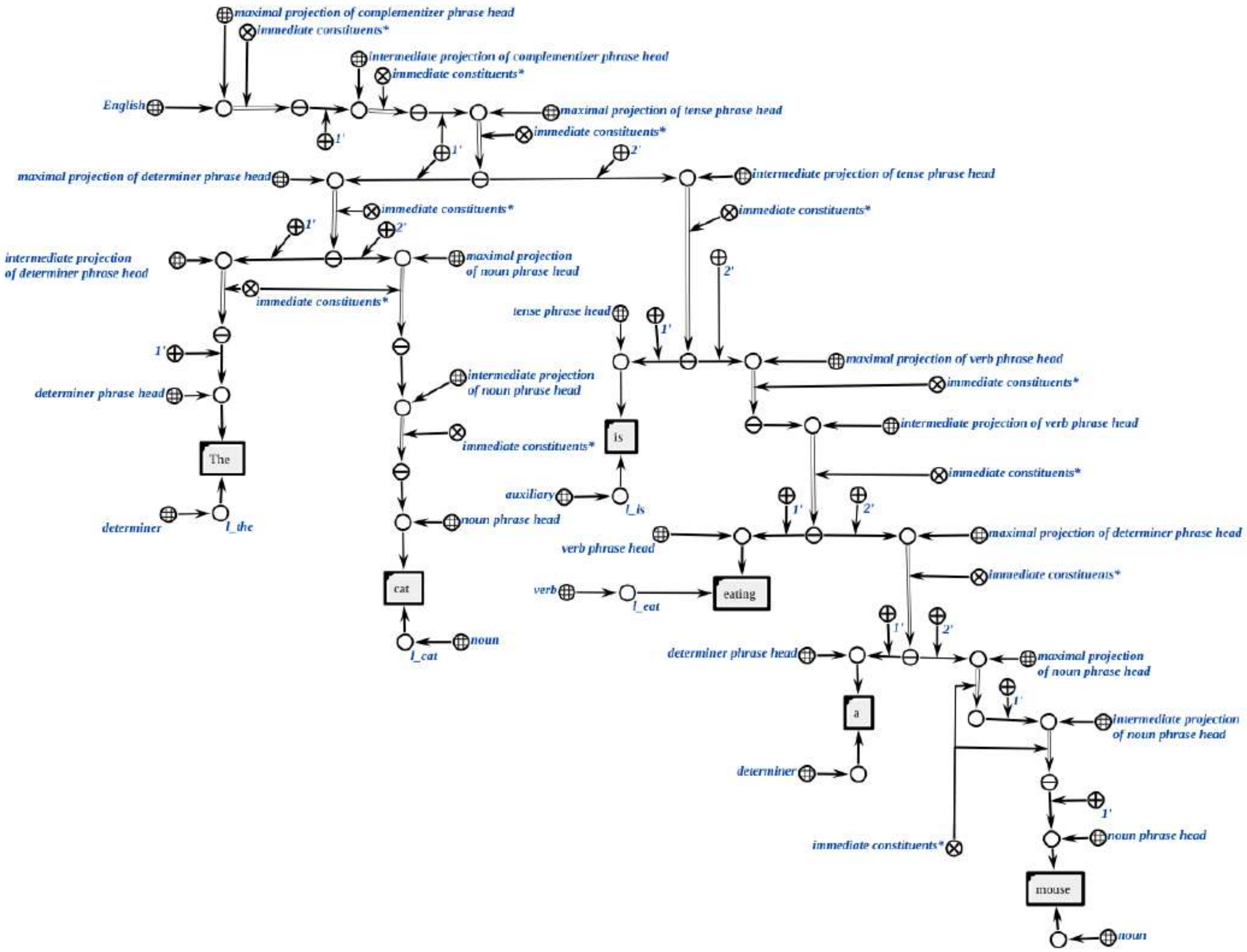

maximal projection of complementizer phrase head
immediate constituents*
intermediate projection of complementizer phrase head
immediate constituents*
English
1'
maximal projection of tense phrase head
immediate constituents*
1'
2'
maximal projection of determiner phrase head
intermediate projection of tense phrase head
immediate constituents*
immediate constituents*
1'
2'
intermediate projection of determiner phrase head
maximal projection of noun phrase head
2'
immediate constituents*
tense phrase head
1'
1'
maximal projection of verb phrase head
immediate constituents*
intermediate projection of noun phrase head
determiner phrase head
intermediate projection of verb phrase head
The
is
immediate constituents*
immediate constituents*
auxiliary
l_is
1'
2'
determiner
l_the
noun phrase head
verb phrase head
maximal projection of determiner phrase head
immediate constituents*
verb
l_eat
eating
cat
1'
2'
noun
l_cat
determiner phrase head
maximal projection of noun phrase head
1'
a
intermediate projection of noun phrase head
determiner
1'
immediate constituents*
noun phrase head
mouse
noun

Figure 5. Syntactic structure example.

Dialogue context that does not change during system operation contains the knowledge necessary to ensure that the system performs its functions, which were put into it a priori by its developers and/or administrators and do not change during its operation on an ongoing basis.

Dialogue context that changes during system operation contains the knowledge necessary for the system to perform its functions, which were obtained during its operation and/or the validity of which is short-lived.

dialog context that changes during system operation
⇒ *subdividing**:
Typology of contexts that change during system operation knowledge source^
= {• *dialog context containing knowledge from external sources*
• *dialog context containing knowledge gained during a dialog*
}

⇒ *subdividing**:
Typology of changing contexts according to their validity^
= {• *valid dialog context*
• *invalid dialog context*
}

A subset of the context can be included in the approved part of the KB, for example, if we are talking about some biographical information previously entered in the KB - date of birth, etc.

At a given moment, one user dialog context is associated with a user (containing at minimum facts about them known in advance: name, age, etc.) and several thematic contexts. A context specification example is shown in figure 9.

Thus, **action. context resolution** is reduced to mapping each meaning variant to the corresponding context. The choice is made on the basis of the value of the function $F_{CTD}(T, C)$, where T is a translation variant, C is a thematic context. A suitable context for a translation variant is the one for which

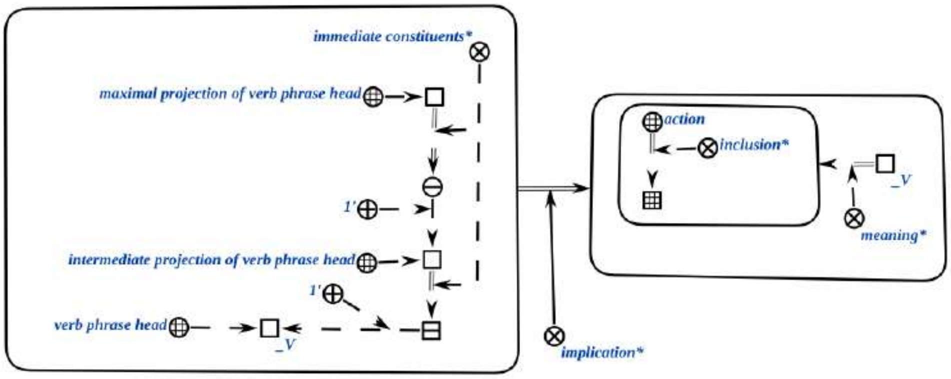


Figure 6. An example of transitioning from syntactic structure to its semantics.

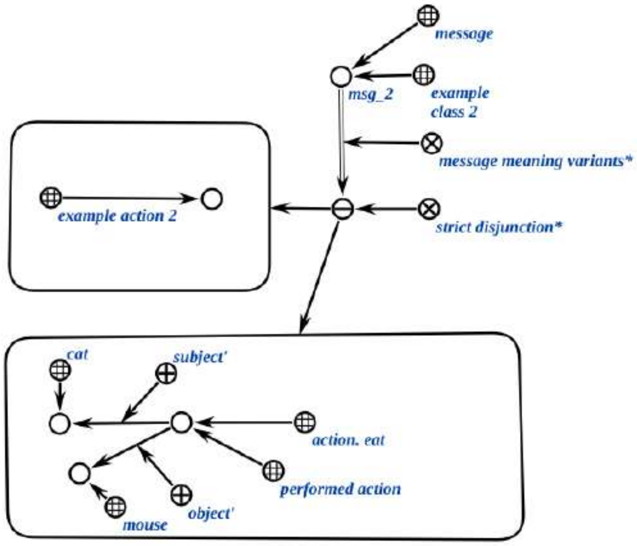


Figure 7. An example of a construction specifying potential meaning of a message.

the value of this function is the highest. If a suitable context is not found, a new one is generated. An example of the result of this action is shown in figure 10.

Action. choosing the meaning of the message is the choice from a set of translation options and their corresponding contexts of one pair and its designation as a construction equivalent to the message. In the simplest case, at this stage, it is permissible to make a choice in accordance with the values of the $F_{CTD}(T, C)$ function calculated at the previous stage for pairs of potentially equivalent structures and their corresponding contexts and choose the pair for which it has the highest value, but if necessary, it is also possible to introduce a separate function. An example of the result of this action is shown in figure 11.

Action. embedding of a message into a context is the embedding of the resulting meaning of a message into a context. In addition to the chosen meaning of the message, other information necessary for processing the message can be added to the context. Moreover, at this stage of the analysis, pronoun resolution should also be performed based on the

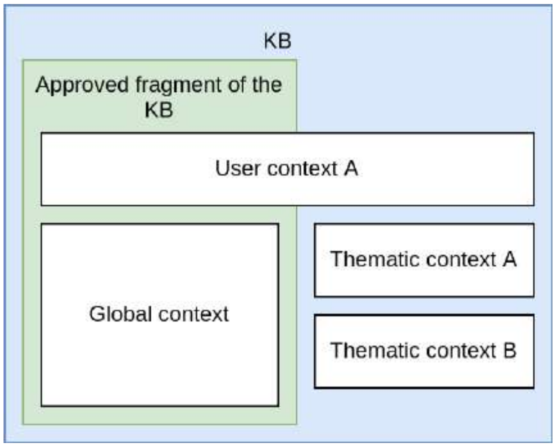


Figure 8. Relationship of contexts with the approved part of knowledge bases.

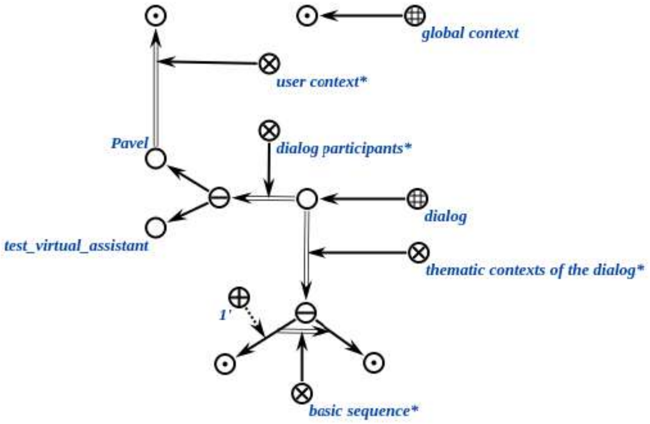


Figure 9. Context specification example.

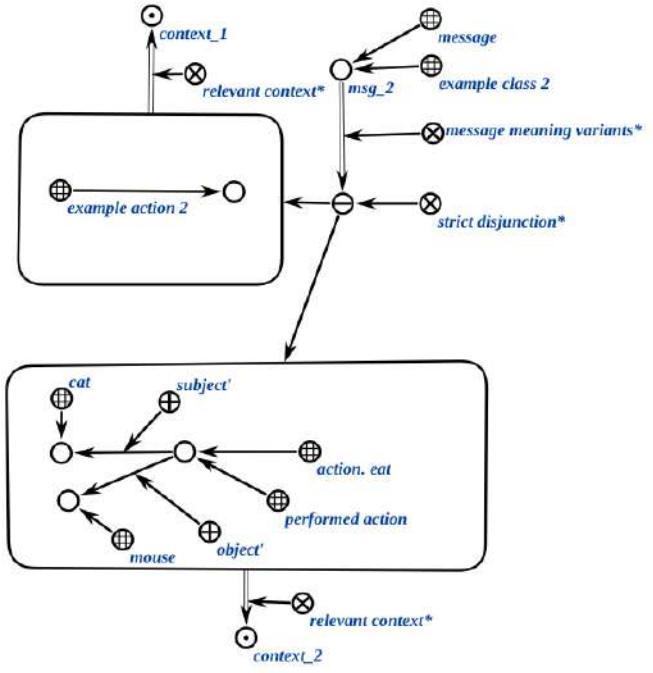


Figure 10. An example of a message with context associated with all meaning variants.

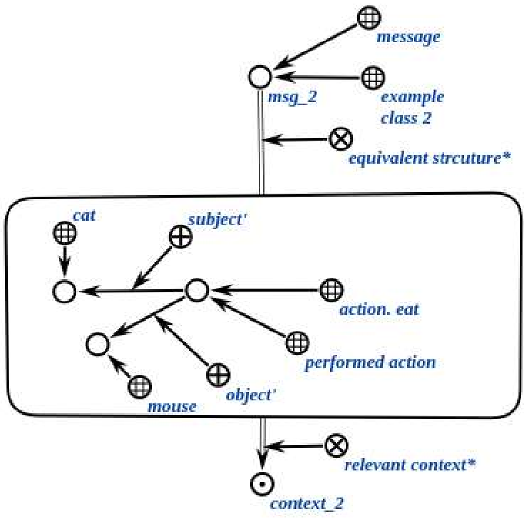


Figure 11. An example of a construction describing a structure equivalent to a message.

information stored in the context. Examples of contexts before and after the message is embedded in it are shown in figures 12 and 13.

Thus, relevant information is collected in a thematic context, and by combining it with the user context and the global context you can get a common context based on which the required system actions should be performed, including the generation of a system response.

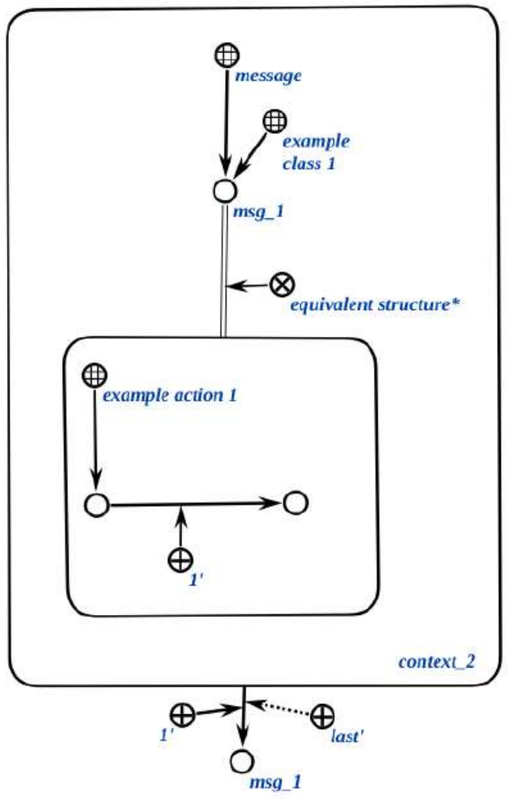


Figure 12. An example of a context before a message is embedded.

IV. Conclusion

An approach has been suggested for the implementation of natural language interfaces of next-generation intelligent computer systems built using OSTIS technology based on a formal model of a natural language, and a dialog context model was introduced.

In the suggested approach, all stages of analysis, including lexical, syntactic and semantic analysis, can be performed directly in the knowledge base of such a system. And all the results (both intermediate stages and the final one) are presented in a single unified form, which helps to ensure compatibility and reduce overhead costs for integrating a subsystem based on this approach.

The formalization of the language used in the basis of this approach is universal and extensible, which makes it possible to supplement it with a formalized specification of a given natural language to ensure the possibility of working with it.

Acknowledgment

The authors would like to thank the research teams of the Department of Intellectual Information Technologies of BSUIR and the Department of Theory and Practice of Translation #1 of MSLU for their help and valuable comments. This research was partially supported by the BRFFR (BRFFR-RFFR No. F21RM-139)..

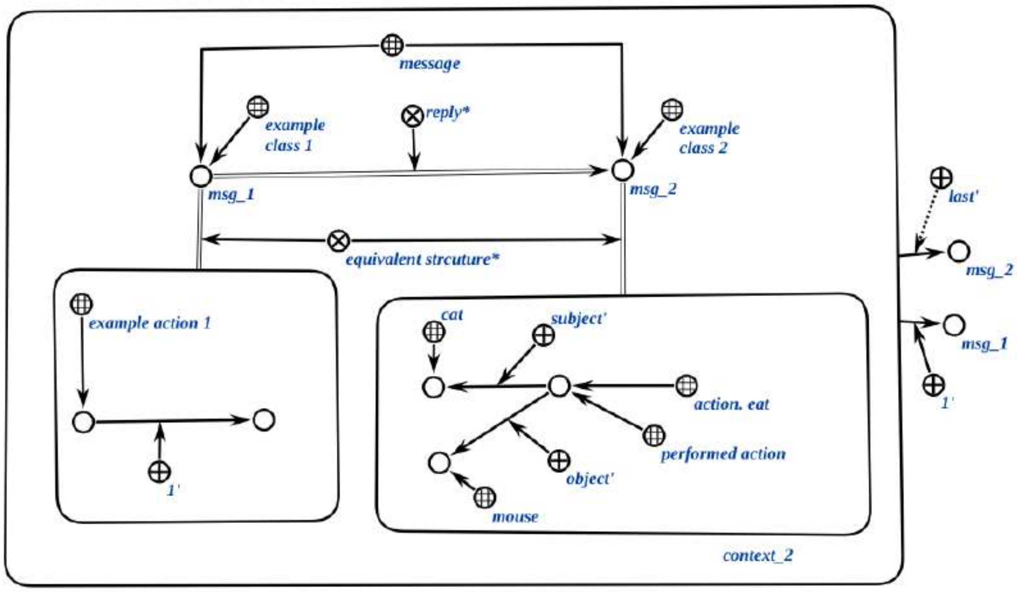


Figure 13. An exaple of a context after a message has been embedded.

References

[1] “Global Voice Assistant Market By Technology, By Application, By End User, By Region, Competition, Forecast & Opportunities, 2024,” Available at: https://www.businesswire.com/news/home/20190916005535/en/Global-Voice-Assistant-Market-Projected-Grow-1.2, (accessed 2019, Dec).

[2] S. Pais, J. Cordeiro, and M. L. Jamil, “Nlp-based platform as a service: a brief review,” *Journal of Big Data*, vol. 9, 04 2022.

[3] D. Trajanov, V. Trajkovski, M. Dimitrieva, J. Dobreva, M. Jovanovik, M. Klemen, A. Žagar, and M. Robnik-Sikonja, “Survey of nlp in pharmacology: Methodology, tasks, resources, knowledge, and tools,” 08 2022.

[4] Khurana, D., Koli, A., Khatter, K. et al., “Natural language processing: state of the art, current trends and challenges,” in *Multimed Tools Appl*, 2022.

[5] E. Strubell, A. Ganesh, and A. McCallum, “Energy and policy considerations for deep learning in nlp,” *arXiv preprint arXiv:1906.02243*, 2019.

[6] “Large language models: A new moore’s law?” Available at: https://huggingface.co/blog/large-language-models, (accessed 2022, September).

[7] “Amazon Alexa Official Site: What Is Alexa?” Available at: https://developer.amazon.com/alexa, (accessed 2022, September).

[8] “Siri,” Available at: https://www.apple.com/siri/, (accessed 2022, September).

[9] “Google Assistant, your own personal Google,” Available at: https://assistant.google.com/, (accessed 2022, September).

[10] “Cortana helps you achieve more with less effort. Your personal productivity assistant helps you stay on top of what matters, follow through, and do your best work.” Available at: https://www.microsoft.com/en-us/cortana, (accessed 2022, September).

[11] M. Hoy, “Alexa, siri, cortana, and more: An introduction to voice assistants,” *Medical Reference Services Quarterly*, vol. 37, pp. 81–88, 01 2018.

[12] V. V. Golenkov, N. A. Gulyakina, D. V. Shunkevich, *Open technology for ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, G. V.V., Ed. Minsk: Bestprint, 2021.

[13] V. V. Golenkov, “Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems,” in *Software products and systems*. Moskow, 2020, pp. 404–412.

[14] R. Jackendoff, *X–bar Syntax: A Study of Phrase Structure*. Cambridge: MIT Press, 1977.

Естественно-языковые интерфейсы интеллектуальных компьютерных систем нового поколения

Гойло А. А. Никифоров С. А.

В данной работе рассматривается подход к реализации естественно-языковых интерфейсов интеллектуальных компьютерных систем нового поколения, построенных по технологии OSTIS, а также предлагается модель контекста диалога. В данном подходе все этапы анализа, включая лексический, синтаксический и семантический анализ могут производиться непосредственно в базе знаний такой системы. Такой подход позволит эффективно решать такие задачи как управление глобальным и локальным контекстами диалога, а также разрешение языковых явлений таких как анафоры, омонимия и эллиптические фразы.

Received 28.10.2022

Ontological Approach to the development of natural language interface for intelligent computer systems

Longwei Qian
Department of Intelligent Information Technology
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
qianlw1226@gmail.com

***Abstract*—Natural language interfaces are oriented towards achievement of interaction between human users and computer systems in the most natural way. In the process of artificial intelligence development, natural language interface has always been the main research direction. The paper is dedicated to the description of the ontological approach to the development of the natural language interface for intelligent computer system (mainly knowledge-based system) based on the OSTIS Technology. In the field of the development of natural language interface the existing approaches to solve the natural language texts processing (natural language texts analysis and natural language texts generation) are considered. Moreover, the ontological model that possibly integrate different kinds of linguistic knowledge in detail and various problem solving models oriented on solving natural language texts processing in a semantically compatible are described.**

***Keywords*—OSTIS, ontology, knowledge-based system, natural language interface, knowledge-driven, knowledge base**

I. Introduction

In the field of artificial intelligence research, recently knowledge-based intelligent systems are widely applied in various fields (e.g. in medical, in education, etc.). As one of the key components of this type of intelligent systems, intelligent user interfaces aim to enable more intelligent interactions between human users and intelligent systems.

In traditional computer systems, the user interfaces in general must have database access capabilities. In accordance with the various means of interaction used by human users, in traditional computer systems, user interfaces usually can be divided into the following types:

- user interfaces with graphical query;
- user interfaces with formal languages query;
- user interfaces based on filling in request forms.

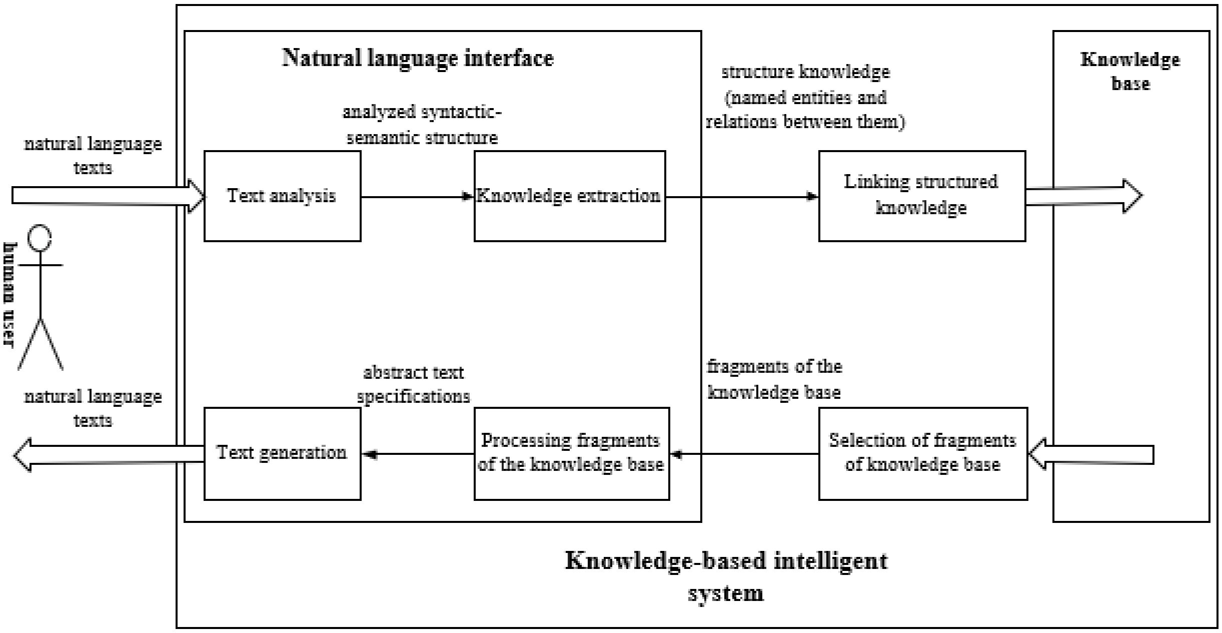
For user interfaces with graphical query, information is exchanged between computer systems and human users through the graphical components (e.g. menus, controls) in the interfaces. For user interface with formal languages query, information is exchanged between computer systems and human users by writing commands in formal languages. Graphical components or commands in formal languages are generally complex and unnatural for the human users. Users must understand the functions of each graphical component in interface or learn the grammar of a certain formal language in advance. For user interfaces based on filling in request forms information is exchanged between computer systems and human users through the completion of specific forms. For form-based interfaces, users need to fill out the forms correctly according to specific requirements. For user interfaces of traditional computer systems, users require additional training (understanding graphical components, learning specific formal language) to interact with computer systems. Therefore human users expect to use more convenient interactive ways without learning complex operations. The natural language interfaces are closest to the natural form of human communication, and can provide human users with more natural interactive experience.

The natural language interfaces of knowledge-based intelligent systems are oriented to achievement of information exchange between natural language texts (especially declarative sentences) provided by human users and knowledge base of intelligent systems in which store specific knowledge. The modular scheme (Figure 1) shows automated information interaction process between human users and intelligent systems based on the classical architecture of modern question-answer systems based on the knowledge base.

As can be seen from the Figure 1, the development of the natural language interfaces of the knowledge-based intelligent systems mainly involves the realization of the following two components:

- conversion of input natural language handwritten texts into knowledge base fragments of intelligent systems;
- generation of natural language handwritten texts from the knowledge base fragments of intelligent systems.

Both conversion of natural language handwritten texts



Natural language interface
Knowledge base
structure knowledge (named entities and relations between them)
analyzed syntactic-semantic structure
natural language texts
Text analysis
Knowledge extraction
Linking structured knowledge
human user
abstract text specifications
fragments of the knowledge base
natural language texts
Text generation
Processing fragments of the knowledge base
Selection of fragments of knowledge base
Knowledge-based intelligent system

Figure 1: Modular scheme of natural language interface of knowledge-based intelligent systems

and generation of natural language handwritten texts are considered subtasks of natural language processing. From the point of view of intelligent systems, these subtasks are kinds of so-called complex problems, the solution of which are still hot research topics. In general, solution of these subtasks requires a combination of various types of knowledge about linguistics and the use of various problem solving models for natural language processing. However in modern intelligent systems, there is a lack of a unified basis to integrate various types of knowledge about linguistic and problem solving models in the development process of natural language interfaces.

II. CLASSIFICATION OF NATURAL LANGUAGE INTERFACE

The development of natural language interfaces for knowledge-based intelligent systems generally requires consideration of following two aspects:

- types of processed natural language, i.e. characteristics of different types of natural language;
- the range of the knowledge base of intelligent systems, i.e. the breadth of knowledge in the knowledge base of intelligent systems.

According to statistics, on the Internet there are only dozens of natural languages that are widely used by human users for communication, such as Russian, English, Chinese, etc. Each natural language has its own unique features, for example, the writing habits of Chinese language are almost completely different from European languages. Therefore for processing various natural language texts, it is necessary to take into account the corresponding characteristics of different natural languages. In addition, the range of the knowledge base also affects the complexity of developing natural language interfaces of knowledge-based intelligent systems. The world-wide knowledge bases [1], [2], in which knowledge sources are oriented to the whole Internet, i.e., commonsense knowledge [3] is stored in the knowledge bases of intelligent systems. The knowledge bases about specialized domain [3], [4], in which knowledge sources are oriented to various encyclopedias (for example, an automobile encyclopedia, an encyclopedia about films, an encyclopedia about various disciplines (such as discrete mathematics, history, and others)), i.e. in the knowledge bases of intelligent systems stores specific industry knowledge (i.e., knowledge about a limited domain). The commonsense knowledge base draws attention to the breadth of stored knowledge, emphasizes the integration of more concepts, named entities across different domains and the relationships between them. from another aspect, the commonsense knowledge base will not provide high accuracy of the knowledge stored in it. With the process of integrating more concepts, named entities and relationships between them in the knowledge base it will lead to higher complexity of searching and processing fragments of the knowledge base. Unlike the commonsense knowledge base, knowledge bases about specialized domain draws more attention to the depth and accuracy of the stored knowledge that satisfies the solution of various specific tasks in intelligent systems with lower complexity.

In accordance with the two aspects listed above, we divide the natural language interfaces of knowledge-based intelligent systems into the following classes:

- natural language interface that is independent of the specific natural language and the specific domain. In this case, this kind of natural language interface can process texts of any kind of natural languages, for example, Russian, Arabic, English, Chinese, and so on, as well as process any complex knowledge structure, i.e. knowledge is stored in the world-wide knowledge bases;

- natural language interface that is dependent on the particular natural language, but is independent of the particular domain. In this case, the natural language interface can only analysis texts of certain natural language and process any complex knowledge structure;
- natural language interface that is independent of the particular natural language, but is dependent on the particular domain. In this case, the natural language interface can analyze texts of any kind of natural languages, which describe facts about the particular domain. The knowledge bases about specialized domains are the basis of the knowledge base of the intellectual systems. It should be noted that the description of facts in different domains may special different types of sentences. Therefore the processing of natural language texts describing facts in different domains (for example, the historical texts, the legal texts, the texts in different disciplines) has the different degree of complexity;
- natural language interface that is dependent on the particular natural language, and as well as is dependent on the particular domains. This kind of interface is considered the most basic natural language interface in knowledge-based intelligent systems. In this case, the natural language interface only needs to analysis texts of certain natural language and process the simpler knowledge structure in knowledge base about the specialized domain.

The development of the natural language interfaces for knowledge-based intelligent systems in a multilingual field requires the study of the structure (in particular, in the syntactic level and the semantic level) of the various natural language texts. Everyone knows that development of world-wide knowledge base is a huge undertaking. As well as world-wide knowledge base is mainly used in Internet-oriented searches, recommendations and question-answer tasks. The application scenario is relatively simple. However, the knowledge density in the knowledge base about the specialized domain is higher, and more complex reasoning and application scenarios can be performed in this kind of knowledge base. The application scenarios of knowledge bases about the specialized domains are more extensive.

The main goal of this article is to use ontological approach as a basis to develop a unified semantic model for natural language interface of knowledge-based intelligent systems. This kind of natural language interface can work in a multilingual field, as well as the knowledge base about the specialized domain are the main components of these intelligent systems, i.e., the natural language interface of intelligent systems that is independent of the particular natural language, but is dependent on the particular domain. The proposed mechanism allows, in principle, to potentially implement information transformation between texts of various natural languages and the knowledge base of intelligent systems about specialized domain, it's just that the complexity of construction of knowledge base about the specific natural language that provides linguistic knowledge of the specific natural language for natural language text processing will be different depending on features of a particular natural language, as well as specific additional problem solvers will be required for solving tasks in the specific natural language text processing.

III. REVIEW OF EXISTING APPROACHES

In the development of the natural language interfaces of knowledge-based intelligent systems, currently two main historically established directions [5] can be distinguished to solve the problem of converting natural language texts into fragments of the knowledge base of intelligent systems, and the problem of generating natural language texts from fragments of the knowledge base of intelligent systems.

The first direction involves developing the systems of logical reasoning based on rules that correspond to the grammar of a certain natural language, formulated by computational linguists and other linguists.

From the point of view of natural language processing, the problem of converting natural language texts into fragments of the knowledge base of intelligent systems is considered as the knowledge extraction. The knowledge extraction refers to obtaining factual information such as concepts, named entities, relations between them, as well as a certain type of events from natural language texts, with the extracted results formalized in the form of an internal knowledge representation language of the intelligent systems (such RDF, SC-code and others). In fact, from the point of view of construction of knowledge base of intelligent systems, the relations between concepts or named entities are considered as the special entities that are extracted from natural language texts.

Nowadays knowledge extraction from natural language texts can be divided into two directions according to the scope of the extracted domain [6]:

- knowledge extraction from closed domains;
- knowledge extraction from open domains.

The knowledge extraction from closed domains is focused on the processing of natural language texts that describe a specific subject domain, for example, football, weather, and so on. When solving the problem of knowledge extraction from closed domains, researchers believe that in a certain subject domain there are limited types of concepts, named entities and relations between them. Therefore knowledge extraction from closed domains often requires predefined types of concepts, named entities and relations between them. However, the goal of knowledge extraction from open domains is to extract various sets of concepts, named entities and relations between them from massive and heterogeneous natural language texts corpora without requiring a predetermined vocabulary to define the types of concepts, named entities and relations between them. In other words, for massive and heterogeneous corpora of natural language texts, the types of concepts, named entities and the relations between them can be infinite, unknown, or even constantly changing.

In the early stages of the development of technology for knowledge extraction from closed domains, researchers study the rules that exist in natural language texts and automatically implement these rules when solving the knowledge extraction. The knowledge extraction from natural language texts in a standardized format has been achieved through artificial inductive and generalized patterns and rules. Therefore, rule-based approaches [7] are closely related to specific natural language texts. As soon as minor changes to the format specification of natural language texts occur, the established rules may not apply. The most obvious advantages of knowledge extraction based on handwritten rules are that they do not require a large of training corpus to train the model and high accuracy. However the construction of rules manually is inefficient, and the constructed rules lack portability in other intelligent systems.

In order to solve the knowledge extraction from closed domains, ontology-based approaches use different models for knowledge extraction. Ontology construction is the basis of knowledge extraction, in the processing of ontology construction the different models are used . The ontology can more accurately describe concepts and relationships between them in the specific domains. In the ontology-based approaches knowledge extraction is often implemented by organically combining multiple ontologies. Alexander Schutz et al. organically combined ontologies such as DOLCE, SUMO, SpotEventOntology to describe related concepts in the field of football and the relationships between them, and then established the RelExt system [8], which can automatically identify correlated triples (a pair of concepts and a relationship between them). However in the knowledge extraction systems based on ontologies the the lack of general principles for the implementation of automatic text processing and knowledge extraction based on ontologies in a unified framework leads to the fact that it cannot ensure the compatibility of various components (different ontologies, problem solving models for natural language text processing and knowledge extraction).

With the development of the Internet, the volume of linked electronic natural language texts is rapidly increasing. In heterogeneous massive texts, the types of named entities and the relationships between them are not defined and limited. In order to meet the requirements of practical application in this situation, more and more researchers are beginning to study technologies for knowledge extraction from open domains. The knowledge extraction from open domains directly determines relative words or phrases in the natural language texts by analyzing natural language texts (particularly, natural language sentences) to realize the extraction of named entities and relationships between them without the need to predetermine types of named entities and relationships between them.

Until now some OpenIE systems (knowledge extraction from open domains) have been developed to extract structured knowledge (i.e. named entities and relationships between them) represented in the internal knowledge representation language from English sentences. As the first generation of the OpenIE systems, TextRunner [9], WOE [10] formulated an mode of knowledge extraction from open domains: first, heuristic or supervised distance learning models are used for automatic sentence labeling; then sequence labeling models (for example, sequence to sequence models) are used in order to learn how to automatically extract knowledge from the labeled sentences; finally, the sentences are entered into the systems, and the systems identify the named entities and the relationships between them in the sentences. With the development of knowledge extraction technologies, researchers have proposed the second-generation mode of knowledge extraction system . Fader et al. developed the ReVerb system [11], which uses a common syntax and lexical constraints for knowledge extraction. This system performs a comprehensive morphological and syntactic analysis of randomly selected English sentences, recognizes verb phrases to express as the relationships between named entities in sentences, and the noun phrases or other phrases in the sentences related to the verb phrases are expressed as named entities. In order to extract more than just verbs or verb phrases from sentences as relations, Mausam et al. proposed the OLLIE system [12] for extracting phrases other than verb phrases as relations with the help of dependency parsing trees of input sentences.

As seen, all the OpenIE systems presented above are focused on knowledge extraction from English sentences. For other natural languages that are very different from English, such as Chinese, the structure of the sentences and the grammatical features between the English and Chinese have the huge difference. It is not known whether these technologies and approaches work for English and also for other natural languages. In order to extract knowledge from those natural languages that differ from English, when the development of the knowledge extraction system it's necessary to take into account the characteristics of the specific natural languages. Tseng et al. proposed a classical architecture for knowledge extraction from open domains for Chinese language. Their CORE system [13] is the first attempt to acquire knowledge from open domains for Chinese texts. The system uses a number of Chinese language processing technologies, such as word segmentation, automatic morphological markup (POS tagging) adjusted and appropriate for Chinese texts, syntactic parsing, semantic analysis, and extraction rules for Chinese sentences.

According to the above, when solving the problem of knowledge extraction from open domains, it is necessary to solve a number of subtasks. However, in the existing knowledge extraction systems there is no unified framework that ensures the consistent use of various models to solve these subtasks. In addition, existing knowledge extraction systems cannot provide a single semantic framework that uses different knowledge bases of linguistics to solve the problem of knowledge extraction. In the unified knowledge base describes the syntactic and semantic knowledge, as well as the logical rules that are used to solve the subtasks in the process of knowledge extraction.

For natural language generation there is a classic pipeline architecture. The following three main modules are the basis of the natural language generation systems developed based on the pipeline architecture [14]:

- content planning: specifying what information from the input data will be included in the generated texts, and how it will be organized;
- micro-planning: deciding how the selected information will be implemented in the form of natural language sentences;
- implementation in natural language: producing grammatically correct natural language sentences.

At an early stage in the applied field, many natural language generation systems were successfully developed in the field of journalism and media research, for example, soccer reports [15], weather or financial reports [16], [17], [**?**] and others. These systems widely use templates and grammar rules constructed by researchers to generate natural language texts. When specific application fields are small and changes to the specific data structures are minimal, template-based approaches can achieve good results for natural language generation. However, the developed templates are highly dependent on the specific application fields and specific natural languages, i.e. the templates do not generalize well to different application fields and different natural languages. In addition, manually developing templates is a laborious task (templates construction requires a significant overhead of labor and time), and the re-usability of templates is low.

The NaturalOWL system [19] is a classical natural language generation system that reasonably applies templates and linguistic rules to generate natural language texts that describe fragments of OWL ontologies. NaturalOWL is focused on generating natural language texts (mainly the English texts) from fragments of OWL ontologies. Currently, the so-called semantic reasoners that perform logical inference on ontologies presented in the OWL format can be developed, but most of the developed reasoners are capable of performing only direct logical inference based on the relations described in the ontology. However in the NaturalOWL system, the construction of templates and rules for generating natural language texts does not use the OWL standard. In other words, the semantic reasoners based on description logic cannot be used to construct templates and linguistic rules. It can be said that this system does not provide a model, which allows representing various types of knowledge in a single knowledge base, including linguistic knowledge, inference on the ontologies of linguistics, even the templates developed by ontologies.

The second direction, widely used in modern intelligent systems, involves the use of various machine learning models based on mathematical statistics and information theory, which are aimed at modeling natural language texts.

At present, approaches based on mathematical models are mainly used in knowledge extraction from closed domains. In recent years, some deep learning models have shown significant advantages in natural language processing. These models have also been applied in the field of knowledge extraction, for example, pre-training models series BERT, GPT achieve better results [20], [**?**]. These approaches encode the input natural language texts into the input information (embedding information), and then decode the input information into predefined markup for each word of the input natural language texts. Based on the marked-up natural language texts, the corresponding named entities and the relationships between them can be extracted. Statistical-based approaches reduce the dependence of knowledge extraction systems on humans. However, using the machine learning models or the deep learning models always requires very large corpora that are manually annotated by the human, hence human intervention is required. Moreover the performance of these models highly depends on the quality of the annotating training samples.

For natural language generation, pre-training models series BERT, GPT also widely applied in the natural language generation systems [22]. For these statistics-based approaches obtaining or constructing the high quality datasets is one of the main key tasks of these approaches, as well as one of the key difficulties. In order to generate natural language texts from fragments of knowledge base, for training these models, training data is usually presented as a pair of inputs (fragments of knowledge base) and outputs (natural language texts).

In recent years, in the WebNLG project [23], the main task is to develop neural generators using deep learning models. For the development of neural generators, the project team provides training data, which consists of Data/Text pairs, where the data are the fragments of the knowledge base in the form of RDF extracted from DBpedia [24], and the texts are the specific natural language texts corresponding to the fragments of the knowledge base. Due to the difficulty of constructing high quality consistent training datasets, only English and Russian datasets are currently provided in the WebNLG project. For other natural languages, such as Chinese, there are no high quality consistent training datasets, although there are several well-known knowledge bases in Chinese in the field of Chinese language processing, for example, CN-DBpedia [25], zhishi.me [26] and others, which can provide the fragments of the knowledge base in the form of RDF [27], [28]. However at the same time, high-quality consistent datasets with Data/Text pairs for generating Chinese language texts are difficult to obtain or construct.

In order to reasonably use different problem solving models to solve the corresponding subtasks in the process of generating natural language texts from the fragments of the knowledge base, Amit Moryossef et al. systematically proposed a text generator developed with a modular architecture that separates the generation process into a symbolic text planning focusing on processing of the fragments of knowledge base [29] and a neural generator focusing on realisation of resulted natural language texts. As seen, the proposed hybrid approaches can improve the performance of text generators with the help of modular architecture. However, in the development of the text generators, it is obvious that there is no unified formal basis for integrating various problem solving models when solving the complex task of generating natural language texts from the fragments of the knowledge base.

IV. PROBLEMS, NEED TO BE SOLVED

During the development of natural language interfaces of knowledge-based intelligent system, Whether for converting natural language texts into fragments of the knowledge base of intelligent systems or generating natural language texts from fragments of the knowledge base of intelligent systems, existing approaches only partially solve the problems. As mentioned above the developed knowledge extraction systems for the open domain and the developed various generators has been successfully applied in various fields. Nevertheless, the following problems still need to be considered:

- In modern intelligent systems, the lack of unified basis in the process of developing the natural language interfaces and the integrating various developed components by different developers leads to the impossibility of parallelization of the components development;
- Due to the diversity of natural languages, when solving the knowledge extraction from open domains and natural language generation, there are their own additional problems for the characteristics of different natural languages, for example, according to the writing habits of Chinese texts, Chinese characters are written one after the other, without natural spaces between them. Moreover the syntactic structures and parts of speech used for English or Russian language processing are not fully applicable to Chinese language processing. Therefore in modern intelligent systems the existing component or modular approaches cannot guarantee the compatibility and flexibility of the developed components in natural language interfaces for multilingual field;
- For converting natural language texts into the fragments of knowledge base, existing ontology-based approaches are only applicable to the knowledge extraction from closed domains. In ontology-based approaches, the lack of a unified principle for the organic combination of various ontologies leads to a large number of incompatible components in the knowledge base. In the development of modern extract knowledge systems from open domains, the lack of a unified framework to integrate automatic text processing and the logical models for knowledge extraction leads to low compatibility of the developed components;
- For generating natural language texts from the fragments of knowledge base, developed generators on the basis of the modular architectures can use various problem solving models, but the lack of unified principles for using various problem solving models leads to duplication of similar solutions in different generators. In turn, the overhead costs for the development of generators increase;
- For knowledge extraction from open domains or natural language generation, it is necessary to use various levels of linguistic knowledge, including syntactic knowledge, semantic knowledge, logical rules for knowledge extraction, and also other rules to solve specific subtasks. In the natural language interfaces, generally the knowledge base of linguistics provides the necessary knowledge to solve the corresponding tasks. Among the existing various knowledge bases related to linguistics, the lack of unified principles of combining various types of knowledge and knowledge representation models in a single knowledge base, i.e. the possibility of representing syntactic knowledge, semantic knowledge, logical rules in a single knowledge base is difficult. Therefore, the compatibility of developed components of knowledge base by different developers

cannot be guaranteed. The low re-usability of the developed components leads to a high labor intensity in the development of the knowledge base.

In this paper in order to solve the above problems, it is proposed ontological approach to develop a unified semantic model that provides the unified basis to effectively combine the knowledge base of linguistics, including syntactic knowledge, semantic knowledge and other various types knowledge, and various problem solving models for developing the natural language interfaces, solving the problem of converting natural language texts into fragments of the knowledge base and the problem of generating natural language texts from fragments of the knowledge base. The principles underlying the ontological approach to develop a unified semantic model of natural language interfaces will be discussed below.

V. THE PROPOSED APPROACH

For the implementation of converting natural language texts into fragments of knowledge base and generating natural language texts from fragments of knowledge base in the natural language interfaces of knowledge-based intelligent systems, we propose to use ontological approach to represent various linguistic ontologies in a single knowledge base, including syntactic, semantic knowledge and logical rules for texts processing and to develop problem solvers having ability to integrate various approaches for solving related problems in the natural language interfaces within OSTIS Technology framework [30]. The OSTIS technology is aimed at developing a class of systems called knowledge-driven computer systems. Within OSTIS Technology framework this kind of knowledge-driven computer systems developed using the OSTIS Technology are called ostis-systems.

Within the OSTIS Technology, as an internal formal language for the semantic representation of knowledge in the memory of ostis-systems, the SC-code is used, which provides a unified version of information encoding and a formal basis for developing model of ostis-systems. This kind of knowledge representation model is a unified semantic network with a set-theoretic interpretation. The alphabet of SC-code consists of five main sc-elements, on the basis of which it is possible to build SC-code constructions of any complexity. Information is stored in the memory of ostis-systems (sc-memory) as SC-code constructions. Several universal variants of visualization of SC-code [30], such as SCg-code (graphic variant), SCn-code (nonlinear hypertext variant), SCs-code (linear string variant) will be shown below.

Within OSTIS Technology framework, each ostis-system consists of a complete model of such a system, described using a platform-independent SC-code (sc-model of a computer system), and an interpretation platform for such an sc-model, which can generally be implemented in software and hardware. From the point of view of components, the sc-model of the computer systems (ostis-systems) can conditionally be decomposed into the sc-model of the knowledge base [31], [32], the sc-model of the problem solver [33], [34], and the sc-model of the interface [35], [36], as well as the model of abstract semantic memory (sc-memory) [37], which stores the constructions of the SC-code (Figure 2). The principles of the development of knowledge bases and problem solvers are discussed in more detail in [31] and [33], respectively. Moreover the principles of the development of sc-model of the user interface, which is included in the sc-model of the interface, were desecribed in the article [35], [36].

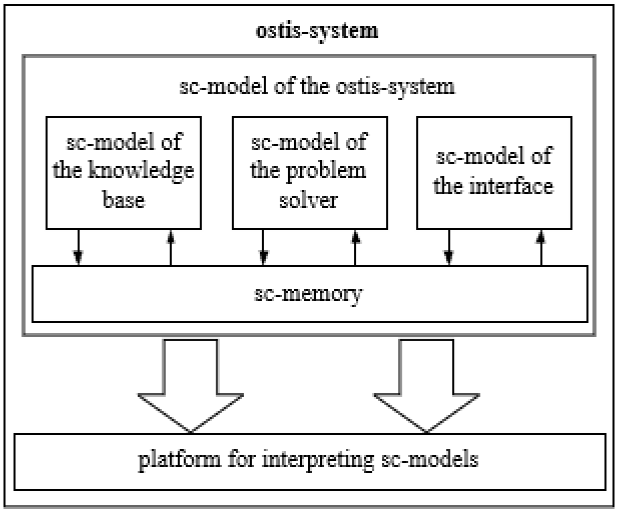

ostis-system
sc-model of the ostis-system
sc-model of the knowledge base
sc-model of the problem solver
sc-model of the interface
sc-memory
platform for interpreting sc-models

Figure 2: The architecture of the ostis-system

Within OSTIS Technology framework, according to the type of interaction between the human users and the computer systems, natural language interfaces of ostis-systems are defined as a subclass of user interfaces [38]. Therefore the development of natural language interfaces of ostis-system in fact is the development of sc-model of natural language interface, which is capable of solving related problems about natural language text processing in the natural language interfaces.

The OSTIS Technology is used as a formal basis to develop the sc-model of natural language interfaces of ostis-systems, the following advantages and characteristics are mainly considered:

- The SC-code is aimed at the sense knowledge representation, which provides a unified semantic model and means for generalizing various types of knowledge and knowledge representation models (for example, the semantic network, the frame, the production rule representation, the logical model) in a single knowledge base;
- The ontological and modular approaches are used to develop the components of the intelligent systems,

which allow integrate various types of linguistic knowledge and various problem solving models for natural language processing in a semantically unified manner. Thence these approaches ensure semantic compatibility and re-usability of the developed components;

- The structure of the knowledge base is developed as a hierarchical system of selected subject domains and their corresponding ontologies. This hierarchical structure allows describing various levels of linguistic knowledge in detail, including syntactic level, semantic level and others. Therefore according to the developed sc-model of knowledge base of the linguistics, for the specific natural language, the structure of the knowledge base is relatively easy to adjust to construct the knowledge base on specific natural language processing (in general, the specification of the syntax and semantics of the specific natural language is only need);
- The multi-agent approach is used to develop the sc-model of problem solver in the intelligent system. This approach allows to consider the interpretation of problem solvers as a hierarchical system of agents (sc-agents) working in a unified semantic memory (sc-memory). The sc-agents are a certain kind of subjects that perform actions in sc-memory. In order to solve problems at different levels on natural language processing this approach provides a unified basis for integrating various problem solving models (for example, logical models and artificial neural network models);
- The multi-agent model for development of problem solver provides developed component oriented on natural language processing modifiability, i.e. the ability to extend functionality of each components or to improve their performance.

Within OSTIS Technology framework, natural language interfaces of the ostis-systems, as a subclass of user interfaces, can be considered as the specialized ostis-systems focusing on solving the problems related to natural language interfaces (concretely, conversion of natural language texts into fragments of knowledge base and generation of natural language texts from fragments of knowledge base). According to the principles to develop the ostis-systems, the development of natural language interface generally includes the development of a knowledge base (mainly the knowledge base of linguistic) and a problem solver of the natural language interface [40]. The knowledge base of natural language interface mainly requires the presence of an sc-model of knowledge base of linguistic, actions for analysis of natural language texts, as well as actions for generation of natural language texts, as shown in Figure 3.

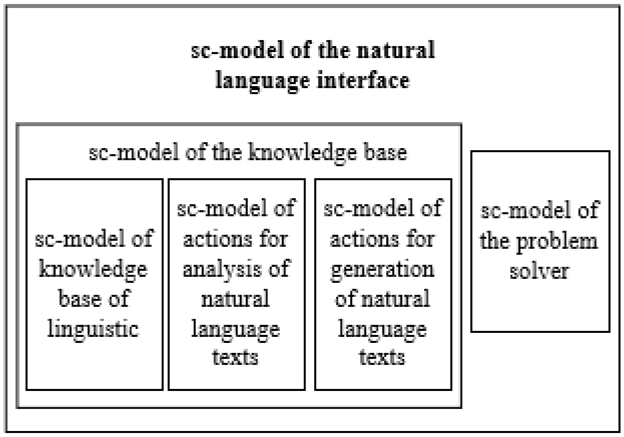

sc-model of the natural language interface
sc-model of the knowledge base
sc-model of knowledge base of linguistic
sc-model of actions for analysis of natural language texts
sc-model of actions for generation of natural language texts
sc-model of the problem solver

Figure 3: The general structure of the sc-model of the natural language interface

As can be seen from the Figure 3, due to the use of component approach, the development of the entire natural language interface comes down to development and improvement of separate specified components. Within OSTIS Technology framework, it is assumed that each problem solving model corresponds to some sc-agent. The knowledge base of linguistic just provides static linguistic knowledge to problem solving model for problem solution. The natural language interface should have the dynamic ability to perform some actions aimed at solving particular related problems in the natural language interfaces.

VI. SC-MODEL OF KNOWLEDGE BASE

According to the basic principles to construct the knowledge base using OSTIS Technology, the development of knowledge base is to consider structure of knowledge base as a hierarchical system of subject domains and their corresponding ontologies. From the point of view of OSTIS Technology, an ontology is interpreted as a specification of the system of concepts of the corresponding subject domain. The knowledge base of linguistic contains a formal description of necessary linguistic knowledge for analysis of natural language texts and generation of natural language texts [5]. The corresponding ontology provides a formal description of the concepts used for the representation of such linguistic knowledge in the different levels from basic word to syntactic and semantic structure of natural language texts. Linguistic theories offered by linguists provide a theoretical basis for building knowledge base of linguistics. Therefore the main hierarchy of knowledge base of linguistics used in the natural language interfaces to solve the particular problems related directly to conversion of natural language texts into fragment of knowledge base of ostis-systems and generation of natural language texts from fragment of knowledge base of ostis-systems is shown below in the SCn-code.

knowledge base of Linguistics
⇐ *section decomposition**:
{• *Section. Subject domain of lexical analysis*
• *Section. Subject domain of syntactic analysis*
• *Section. Subject domain of semantic analysis*
}

A. subject domain of lexical analysis

The subject domain of lexical analysis includes a series of ontology for lexical analysis, which describes characteristics of words and syntactic function of words, parts of speech, etc. A fragment of the ontology of the subject domain is presented below:

natural language text
⊂ *file*

word
⊂ *file*

nominative unit
⊂ *file*

The *word* is the smallest of the natural language units, carrying semantics, which serves to name objects, their qualities, characteristics and interactions, as well as for service purposes. A *nominative unit* is a stable string of word combinations.

natural language text
⇒ *decomposition**:
{• *part of speech*
⇒ *decomposition**:
{• *content word*
• *function word*
• *modal word*
}
• *sign of syntax alphabet*
}

The *part of speech* is a category of natural language words determined by morphological, syntactic and semantic features. The *signs of the syntax alphabet* are auxiliary syntactic means (at the macro level - prepositions, postpositions, conjunctions, particles, etc., as part of service words, at the micro level - inflections, prefixes, postfixes, infixes, punctuation, etc.) [**?**].

It is important to note that these ontologies in the respective subject domain are generally recognized in most natural languages. In other words, due to the diversity of natural language, the linguistic theory of a certain natural language is not suitable for other natural languages. Thus, when developing the knowledge base of linguistics for a particular natural language, the specification of subject domains must take into account the specific characteristics of a particular natural language.

In Russian, English or other European languages, word structure is studied within the framework of morphology. It's necessary to consider the relation *morphological paradigm**:

lexeme
⊂ *file*
⇒ *explanation**:
[The lexeme is a unit of lexical meaning that underlies a set of words that are related through inflection.]

morphological paradigm*
∈ *quasi-binary relation*
⇒ *first domain**:
lexeme
⇒ *second domain**:
word form

As you can see, the terminology lexeme is studied only to describe the features of the words in European languages that have the feature of morphological paradigm.

B. subject domain of syntactic analysis

The subject domain of syntactic analysis describes the characteristics of natural language syntax, the functional characteristics of syntactic components (such as, word, phrase, sentence, etc.). Among them, in the filed of natural language processing sentences have always been considered as the smallest research units. The analysis of sentence is an important intermediate stage, connecting the analysis of the entire text and the analysis of individual words. A fragment of the ontology of the subject domain is presented below in the SCn-code:

sentence
⇒ *decomposition**:
{• *simple sentence*
⇒ *decomposition**:
{• *sentence with subject and predicate*
• *sentence without subject and predicate*
• *special sentence*
}
• *complex sentence*
}

A *sentence* is considered simple if it contains one predicative unit, if more - complex.

part of the sentence'
⇒ *decomposition**:
{• *principal part of the sentence'*
⇒ *decomposition**:
{• *grammatical subject'*
• *grammatical predicate'*
• *grammatical direct object'*

}
- *subordinate part of the sentence'*
  ⇒ *decomposition**:
  {• *grammatical indirect object'*
  • *grammatical attribute'*
  • *grammatical circumstance'*
  }

}

A *part of the sentence* is a role relation that indicates a certain of syntactic relation between the sentence and its components (words, phrase etc.) [39].

relation of dependency*
∈ *non-role relation*
⇒ *decomposition**:
{• *subject**
• *object**
• *complement of verb**
• *coordinate**
• *attribute**
• *adverbial**
}

On the basis of theory of dependency grammar, a *relation of dependency** is a relation that indicates directed link within decomposed two components of sentence, the components are separately called the *head*, which plays the role *grammatical predicate'* in the sentence, and the *depedent*. Each *depedent* has a syntactic function depending on the *head* in sentence. Hence in the syntactic analysis of sentences, the *predicate* of sentence is generally considered as the structural center of the syntactic structure of sentence. All other syntactic components (words or phrases) are directly or indirectly connected with the *predicate* through directed links (i.e., the direction with the *predicate* to other syntactic components).

In the logical ontology of subject domain of the sentence, it is possible to describe a series of logical definitions and logical statements for the sentences. In the form of logical statements, templates or rules about the sentence analysis can be constructed, as well as can be used by problem solvers to solve specific problems in the natural language interfaces.

In the Figure 4 the logical statement in SCg indicates that a sentence with the subject and the predicate can consist of the noun phrase and the verb (or verb phrase), which serve as subject and predicate in the sentence, respectively. As shown Figure 4, all used concepts and relations, such as “sentence with subject and the predicate”, "grammatical subject'”, “grammatical predicate'”, and others, are contained in the subject domain of the sentence. Various type of knowledge built in the subject domain can be used to process natural language texts.

This fragment of the knowledge base describes that the *sentence with the subject and the predicate* can be considered as a sequence of the noun phrase and the verb phrase (or verb) with the corresponding attributes. Moreover, in the process of text generation, a sentence with the subject and the predicate can be generated according to this given structure.

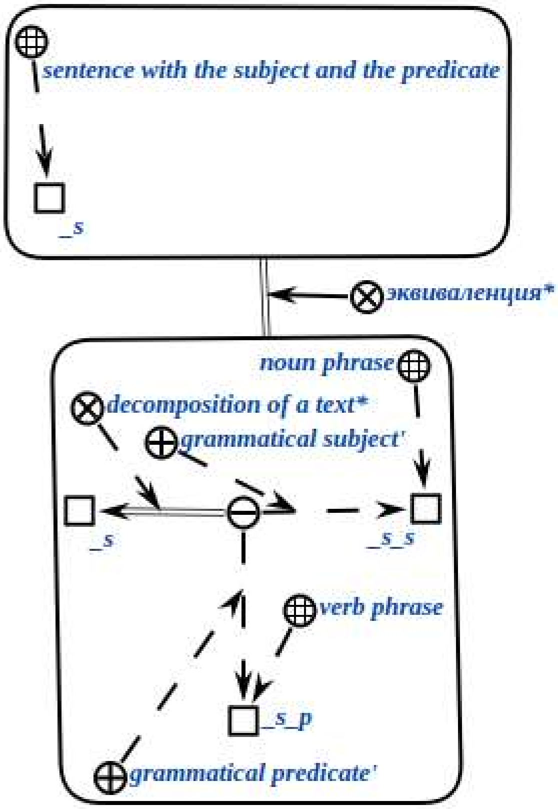


Figure 4: The logical statement about sentence with the subject and the predicate

C. subject domain of semantic analysis

The subject domain of semantic analysis describes the semantic characteristics of the words and the semantic structure of natural language sentences, the functional characteristics of semantic of components (words, phrase and sentence, etc.), the semantic role, the rules for semantic analysis, and so on. The general structure the subject domain is presented below in the SCn-code:

The subject domain of semantic analysis at the level of lexeme, the subject domain of semantic analysis at the level of sentence, and the subject domain of semantic analysis at the level of paragraph describe different linguistic knowledge for semantic analysis at different aspects, respectively.

The subject domain of semantic analysis at the level of lexeme describes the semantic classifications of common basic concepts expressed by the lexeme or the binding of the lexeme to individual entities. The subject domain was constructed based on various types of knowledge bases of semantic about natural language used for semantic analysis at the level of lexeme, for example, WordNet, ConcetpNet, The Semantic Knowledge base of Modern Chinese [41], TAPAZ-2.

semantic of lexeme
⇒ *decomposition**:
  {• *semantic of verb*
  • *semantic of noun*
  • *semantic of adjective*
  • *semantic of adverb*
  }

participant of the exposure*
:= [participant of the action*]
∈ *non-role relation*
⇒ *first domain**:
  individ
⇒ *second domain**:
  action
⇒ *decomposition**:
  {• *subject**
    ⇒ *decomposition**:
      {• *initiator**
      • *inspirer**
      • *spreader**
      • *creator**
      }
  }

An *individ* is a kind of the pattern as a separate entity, which is an instance of the specific concepts.

The *participant of the action** is a non-role relation that connects the action with the individ participating in it, to a certain extent, it can be considered as the semantic role of the action, which generally is expressed by the verb or the verb phrase in the sentence.

In turn, similarly the subject domain of semantic analysis at the level of sentence describes the semantic structure of sentences, the semantic relations between the components (words, phrase and so on) in a sentence, and the semantic relations between the sentences in the text. Some open sources, for example, TAPAZ-2, PropBank and others, served as the basis for building this subject domain.

D. *subject domain of actions for natural language interface*

Previously we consider the subject domains and the corresponding ontology in the knowledge base of linguistic, within which the various type of linguistic knowledge is described. However, the formalization of the linguistic knowledge is not enough for the natural language interfaces to solve the particular problems, the natural language interfaces should perform some *actions* to implement the conversion of natural language texts into fragments of knowledge base and the generation of natural language texts from fragments of knowledge base.

Let us consider the hierarchy of classes of actions for natural language interface in the SCn-code:

action for natural language interface
:= [action for processing the natural language texts]
⊂ *action*
⇒ *decomposition**:
  {• *action for converting natural language texts into fragments of knowledge base*
  • *action for generating natural language texts from fragments of knowledge base*
  }

action for converting natural language texts into fragments of knowledge base
⊂ *action for processing the natural language texts*
⇒ *decomposition**:
  {• *action for decomposing texts into separate units*
  • *action for marking up separate units*
  • *action for syntactic analysis*
  • *action for semantic analysis*
  • *action for determining the knowledge structure*
  • *action for linking knowledge structures in the knowledge base*
  • *action for determining contradictions*
  • *action for resolving contradictions*
  }

action for generating natural language texts from fragments of knowledge base
⊂ *action for processing the natural language texts*
⇒ *decomposition**:
  {• *action for determining the converted knowledge structures*
  • *action for converting knowledge structures into standard basic sc-constructions*
  • *action for determining candidate basic sc-constructions to be generated to texts*
  • *action for converting sc-constructions into message triples (subject-action-object)*
  • *action for generating resulted texts from the message triples*
  }

From the perspective of text generation for fragments of knowledge base of ostis-system, it is worth noting that message triple is a kind of ordered triple represented in the form of <semantic subject, action, semantic object>, where semantic subject is always the identifier of the set of sc-node denoting concepts that are not relations or the element of this set of sc-node; relation is the identifier of the set of sc-node denoting role relation or no-role relation, which points the linking (bundle) that connects the semantic subject and object; semantic object is the identifier of the set of sc-node denoting concepts that are not relations or the element of this set of sc-node.

The message triple is considered as intermediate conversion between the sc-structure and resulted text, mainly because this kind of triple is easier to express as natural language text (mostly sentence) using either rule-template models or modern statistical models. Moreover the message triples are easily to be converted to RDF triples, in turn, can be generated to natural language text using modern deep learning models. Within the framework of OSTIS Technology, the relations defined in the IMS-system are consider as the domain-independent relations, in addition, there are many relations that are dependent on the specific ostis-systems in various subject domain. According to the category of relations, it is divided into different set of message triples corresponding to various types of relations in different sc-constructions.

Since as a result of the actions for converting natural language texts into fragments of knowledge base and the actions for generating natural language texts from fragments of knowledge base, with the help of the natural language interface, the mutual conversion between the natural language texts and the fragments of knowledge base of the ostis-systems is realized, in turn the information interaction between the human users and the ostis-systems is completed.

VII. SC-MODEL OF PROBLEM SOLVER

Within the OSTIS Technology, in order to realize the natural language interfaces of the ostis-systems it need to have ability to perform the above-mentioned actions. Within the OSTIS Technology framework, an action is defined as a purposeful process carried out by one or more subjects. In this case, the sc-agent is considered as a certain subject capable of performing actions in the sc-memory to solve problems. The multi-agent approach is used as a basis for development of the problem solvers. The interaction of agents will be performed exclusively in the semantic memory (sc-memory), which stores the SC-code constructions. Such approach provides the flexibility and modularity of developed sc-agents, as well as provides the ability to integrate various problem solving models corresponding to these developed sc-agents. In the term of implementation, the programs of each sc-agent can implement logical reasoning based on a hierarchy of statements comprised in the logical ontology, as well as the data-driven algorithms (machine learning, deep learning and so on) in various programming language.

Let us consider the general structure for problem solver of natural language interface in the SCn-code:

Problem solver for natural language interface
⇐ *decomposition**:
{
- *Problem solver for converting natural language texts into fragments of knowledge base*
- *Problem solver for generating natural language texts from fragments of knowledge base*

}

From the point of view of OSTIS Technology, the sc-agents (the copies of the same sc-agent or functionally equivalent sc-agents) may work in different ostis-system, and can be considered physically different sc-agents. Rather than considering properties and typology of sc-agents, it's more rational to focus on classes of functionally equivalent sc-agents, which are called abstract sc-agents. The abstract sc-agents is a certain class of functionally equivalent sc-agents, various items of which can be implemented in different ways to specific problems [33].

Problem solver for converting natural language texts into fragments of knowledge base
⇐ *decomposition of an abstract sc-agent**:
{
- *Abstract sc-agent of lexical analysis*
  ⇐ *decomposition of an abstract sc-agent**:
  {
  - *abstract sc-agent of decomposing texts into separate units*
  - *abstract sc-agent of marking up separate units*

  }
- *Abstract sc-agent of syntactic analysis*
- *Abstract sc-agent of semantic analysis*
- *Abstract sc-agent of extracting knowledge structures into the knowledge base*
  ⇐ *decomposition of an abstract sc-agent**:
  {
  - *abstract sc-agent of determining knowledge structure*
  - *abstract sc-agent of linking knowledge structure into knowledge base abstract sc-agent of determining contradictions*
  - *abstract sc-agent of resolving contradictions*

  }
- *Abstract sc-agent of logical inference*

}

Problem solver for converting natural language texts into fragments of knowledge base is a group of sc-agents that implement the mechanisms of extracting fragments of knowledge base from natural language texts. In principle, this problem solver can potentially extract structured knowledge (generally, the named entities, relations, concepts and so on) from various types of natural language texts into the knowledge base of the ostis-system about a specific subject domain, but the construction of the corresponding knowledge base on the specific natural language processing, as well as within which the rules for processing the corresponding natural language texts and the rules for knowledge extraction will become more complex, and also overhead costs will increase.

Abstract sc-agent of lexical analysis – the agents that implement the mechanisms of decomposition of input natural language texts into separate lexical units, and each separate unit belongs to a specific markup according to a certain of markup standard for the specific natural language. Components (words, phrases, sentences, and others) of natural language input texts can be defined according to a certain standard of a particular natural language.

Abstract sc-agent of syntactic analysis – the agents that implement the mechanisms of constructing the syntactic structure of input natural language texts.

Abstract sc-agent of semantic analysis – the agents that implement the mechanisms of constructing the semantic structure of input natural language texts.

Abstract sc-agent of extracting knowledge structures into the knowledge base – the agents that implement the mechanisms that determine knowledge structures in natural language input texts (i.e., the definition of named entities and relationships between them), and construct the knowledge structures in the form of SC-code into knowledge base of the ostis-system about specific subject domain.

In the process of constructing knowledge structures into the knowledge base, when comparing the knowledge structures extracted as a result of the analysis of natural language input texts with the knowledge stored in the knowledge base of a particular ostis-system, the ***abstract sc-agent for determining contradictions*** implement mechanisms for determining contradictions, for example , for a certain named entity in the input natural language texts, there may be several corresponding instance of concept in the knowledge base.

Abstract sc-agent of resolving contradictions - agents that implement mechanisms that resolves the contradictions in case of detection of contradictions.

Abstract sc-agent of logical inference - agents that implement mechanisms that use logical rules written by means of SC-code for syntactic, semantic analysis and also extracting knowledge structures for natural language texts. Thus, this agent can semantically interact with the agents of syntactic and semantic analysis and the agent of extracting knowledge structures.

In order that the natural language interface can generate fluent natural language texts from fragments of knowledge base of ostis-system about specific subject domain, the problem solver for generating natural language texts from fragments of knowledge base is developed based on the classical pipeline for solving the problem of natural language generation. Although the development of this abstract agent is based on the classical pipeline, the development of composed sc-agents in the problem solver can be flexible by using a multi-agent approach. For the problem of generating texts for a particular natural language, the compositions of the problem solver can be easily modified accordingly.

Let us consider the general structure for problem solver for generating natural language texts from the fragments of knowledge base represented in SCn-language:

Problem solver for converting natural language texts into fragments of knowledge base
⇐ *decomposition of an abstract sc-agent**:
{ • *Abstract sc-agent for content selection*
 ⇐ *decomposition of an abstract sc-agent**:
 { • *Abstract sc-agent determining sc-structure*
 • *Abstract sc-agent dividing determined sc-structure into basic sc-structure*
 • *Abstract sc-agent determining the candidate sc-structures*
 • *Abstract sc-agent transferring candidate sc-structures into message triples*
 }
• *Abstract sc-agent text planning*
 ⇐ *decomposition of an abstract sc-agent**:
 { • *Abstract sc-agent ordering message triples*
 • *Abstract sc-agent ordering entities of a message triple*
 }
• *Abstract sc-agent for micro-planning*
• *Abstract sc-agent for surface realization*
}

Abstract sc-agent for determining sc-structure - the groups of agents that provide the specific fragments of knowledge base (i.e. content) in the form of sc-structures, from which the interface will generate the natural language texts.

Abstract sc-agent dividing determined sc-structures into basic sc-structures – the agents that implement the mechanisms of decomposition of determined sc-structures into basic structures, which can be transferred into message triples.

Sometime not all transferred message triples are useful to the end user. According to specific requirements the interface will finally select among the basic sc-structures the ones to generate natural language texts. ***Abstract sc-agent determining the candidate sc-structures*** implements the mechanism of determining the appropriate candidate sc-structures according to requirements.

Abstract sc-agent transferring candidate sc-structures

into message triples – the agents that implement the mechanism of converting candidate sc-structures into message triples.

The order of transferred message triples, as well as the order of elements in each message triple will completely influence on the resulted generated natural language texts. ***Abstract sc-agent for text planning*** implements the function of the ordering of elements in each message triple and the ordering of message triples themselves.

Abstract sc-agent for miro-planning - the agents that implement the mechanism of transferring message triples to abstract sentence specifications that are varied according to selected methods in the natural language generation systems, for example, the simple text templates with slots, syntactic structures and so on.

Abstract sc-agent for surface realization - the group sc-agents that implement the mechanisms of concatenating the sc-links to generate the resulted texts, i.e. filling the sc-links corresponding to elements of message triples with the appropriate forms (generally in European languages there is morphology for lexical units) of lexical units according to rules, and concatenating them.

It is worth noting that developed abstract sc-agents in the above listed problem solvers are general according to the general process of natural language processing, therefore they are flexible and changeable. For development of natural language interface, which can process the specific natural language, it is possible to adjust and make extensions of the already developed abstract sc-agents. Moreover, in some cases for processing of some specific natural languages it is even necessary to add a new agent or remove (deactivation) of one or more existing agents. With the help of multi-agent approach based on OSTIS Technology to develop the sc-agents, the one of advantages is that the development of each agent is independent of each other, i.e. the adjustment of agents, addition of a new agent, even removal of one or more existing agents usually does not lead to changes in other agents.

VIII. Implementation of Chinese Language interface

With the help of the previously proposed sc-models to development of the natural language interfaces of the ostis-systems, the specific natural language interface of intelligent help systems for various subject domains can be implemented. However, due to the laboriousness and complexity of developing an intelligent help system for a specific subject domain, within the framework of this section, the greatest attention will be paid to the implementation of the prototype of the Chinese language interface of the intelligent help system for discrete mathematics.

According to the above-mentioned sc-model of the natural language interfaces of the ostis-systems, on the basis of the general structure of knowledge base of linguistic and problem solver of natural language interface, the development of Chinese language interface requires the development of knowledge base on Chinese language processing, within which constructed various types of linguistic knowledge about Chinese language processing, as well as the development of problem solver for Chinese language processing, consisting of a group of sc-agents, which has possibility to integrate various problem solving models.

In order to implement the Chinese language interface of intelligent help system for discrete mathematics, it's necessary to implement the conversion of Chinese language texts into fragments of knowledge base of intelligent help system for discrete mathematics and generation of Chinese language texts from fragments of knowledge base of intelligent help system for discrete mathematics. The following examples show the processing stage to implement the two main functions in the Chinese language interface.

A. knowledge extraction from Chinese language texts

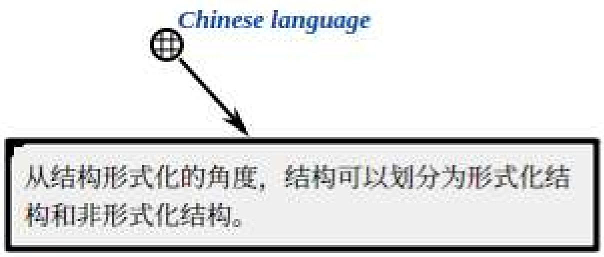
The conversion of natural language texts into fragments of knowledge base is considered as the problem of knowledge extraction. In this case the conversion of Chinese language texts into fragments of knowledge base of intelligent help system for discrete mathematics is mainly to extract named entities and relations between them from Chinese language texts, and the extracted results are stored in the form of SC-code in the intelligent help system for discrete mathematics.

In this example several requirements are defined for processed Chinese language texts:

- The input of Chinese language interface is a standard written Chinese declarative sentence;
- The input Chinese declarative sentence has completed sense and fact information;
- The input doesn't need a predefined vocabulary that includes predefined types of named entities and relations between them;
- The output of Chinese language interface is the sc-structure formally represented in the knowledge base.

From the point of view of OSTIS technology, any natural language text is a file (generally represented as sc-node with content or so-called sc-link). Our example will show the knowledge extraction process on analyzing a Chinese sentence, which is represented in such a node in Fig 5. The content of such a node demonstrates a Chinese sentence that describes: «从结构形式化构角度 (in terms of formalization of the structure) ， (comma) 结构(structure) 可以 (can be) 划分 (divided) 为 (into) 形式化构结构 (formal structure) 和 (and) 非形式化构结构 (informal structure) 。(full stop)».

Step 1 The Chinese sentence is decomposed into separate segmentation units, as well as mark these seg-



从结构形式化的角度，结构可以划分为形式化结构和非形式化结构。

Figure 5: The representation of natural language text in the ostis-system

mentation units with standard part-of-speech categories in Chinese language by agents for lexical analysis (Figure 6).

According to the features of Chinese texts, the Chinese texts consist of a stream of hieroglyphs without natural blanks on the written form. Moreover in Chinese language there are no clear indicators of the categories of number, case and gender, such as in Russian and other European languages, the function of a word in Chinese language becomes clear not on the basis of morphology of a word, but due to its connection with other words. Therefore in the process of analyzing the Chinese texts it is first necessary to perform a lexical analysis, decomposing the stream of hieroglyphs into separate meaningful segmentation units.

Because of the features of Chinese texts, lexical analysis for Chinese texts is more complicated. From point of view of linguists, the definition of Chinese words has always been an important theoretical problem. In order to realize the Chinese language processing by computer, in 1992, the project "The Principle of Segmentation in the Processing of Modern Chinese Information" was released [42], in which the term "segmentation unit" was introduced and a clear criterion of segmentation was established. Moreover in the field of Chinese language processing, the "Modern Chinese word segmentation standard used for information processing" was proposed. In this standard, a word in Chinese language is represented as a "segmentation unit". Its precise definition is "a basic unit for Chinese language processing with certain semantic or grammatical functions. In this case speaking of the Chinese word in this section, we will mean the segmentation unit of the Chinese language.

These decomposed separate units in the lexical analysis meet the principles to be "segmentation units" of Chinese language. Meanwhile the information describing (marking up) these segmentation units satisfying the standards is considered as the linguistic knowledge (e.g. part-of-speech categories) in the lexical analysis of the Chinese texts and is constructed as a part of knowledge base on the Chinese language processing.

Step 2 The agents for syntactic and semantic analysis performs the transition from the lexically marked structure to its syntactic structure or semantic structure, then to the semantically equivalent fragment of knowledge base based on the rules described in the corresponding subject domain.

The main task in this stage is to analyse the relations (or dependency relations) between the input Chinese sentence and separate segmentation units and between these separate segmentation units of that input Chinese sentence to reveal syntactic semantic structure of input sentence. The result of syntactic and semantic analysis is usually a constructed syntax semantic tree or graph, such as phrase structure, dependency structure and others. Based on the constructed syntactic and semantic structure, it is possible to develop a collective of logical reasoning for some applications in the field of Chinese language processing. Therefore the syntactic semantic analysis allows to establish a basis (i.e., a set of logical rules) for extracting structural knowledge from the Chinese sentences. However, the constructed syntactic semantic structure is not conveniently processed in sc-memory. After obtaining the result of syntactic semantic analysis, it is necessary to convert the constructed syntactic semantic structure into a semantically equivalent fragment of knowledge base in the form of SC-code.

From the perspective of the knowledge base, the relations between segmentation units and the Chinese sentence, or the relations between segmentation units of the Chinese sentence, are generally regarded as certain types of linguistic knowledge included in the knowledge base on Chinese language processing.

In the Figure 7 shows a semantically equivalent fragment to the given input Chinese sentences with the help of syntactic semantic structure of this sentence based on the constructed ontologies in knowledge base on Chinese language processing.

The advantage of implementation of transition from input Chinese sentences to semantically equivalent fragments allows to directly use logical reasoning or other knowledge processing operations to analyze the input Chinese sentences based on the dynamic graphical model considered within the OSTIS Technology.

Step 3 knowledge structure is extracted from the semantically equivalent fragment corresponding to the syntactic semantic structure of input Chinese sentence, and fusion of knowledge structure into the knowledge base of intelligent help system (i.e. the construction of fragment of knowledge base) is performed based on the logical rules described in the corresponding subject domain.

When converting Chinese texts into fragments of knowledge base of intelligent help system for discrete mathematics, from the point of view of knowledge extraction from open domain, the main task is to extract named entities and relations between them from the Chinese sentences without requiring predefined types of

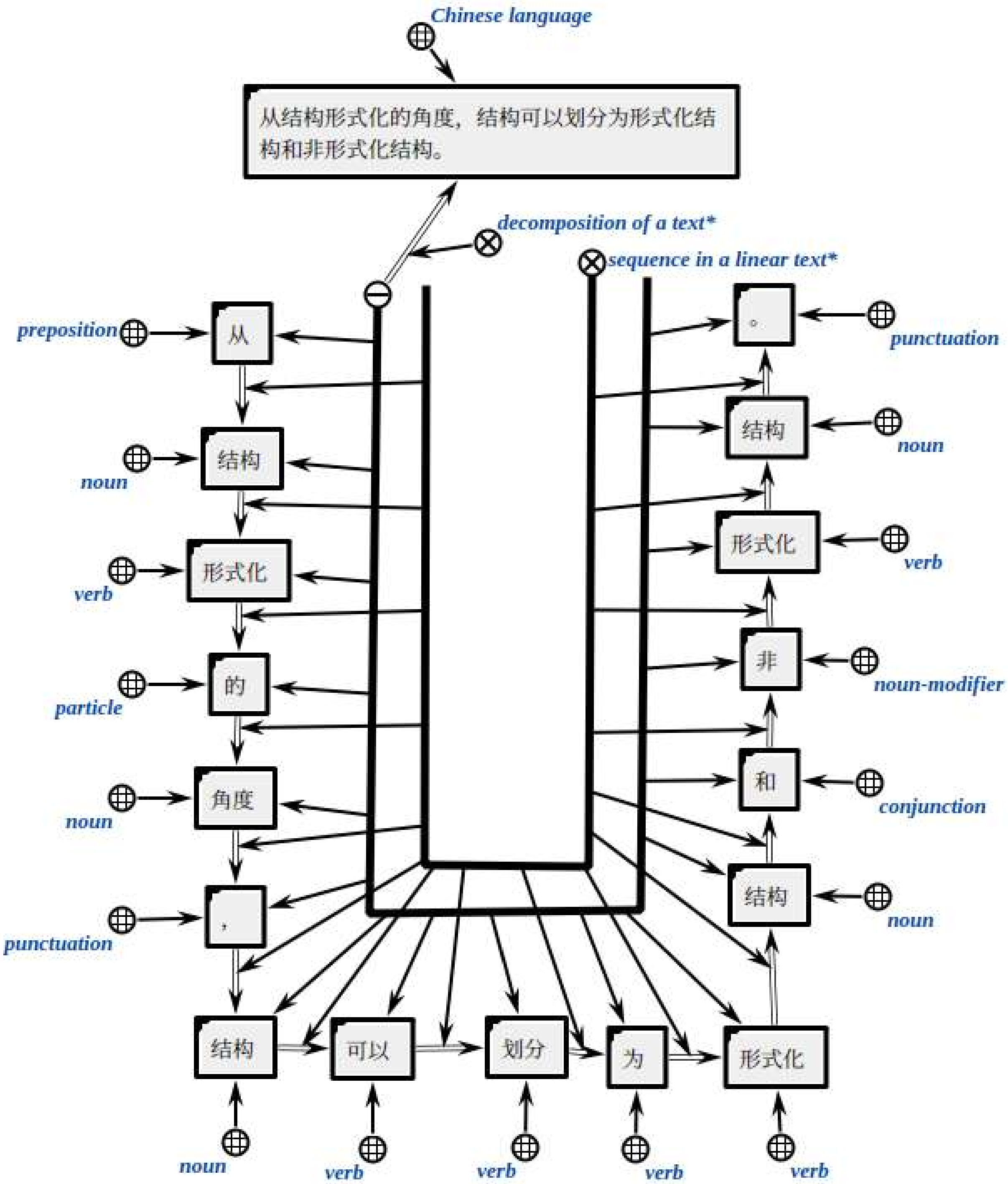
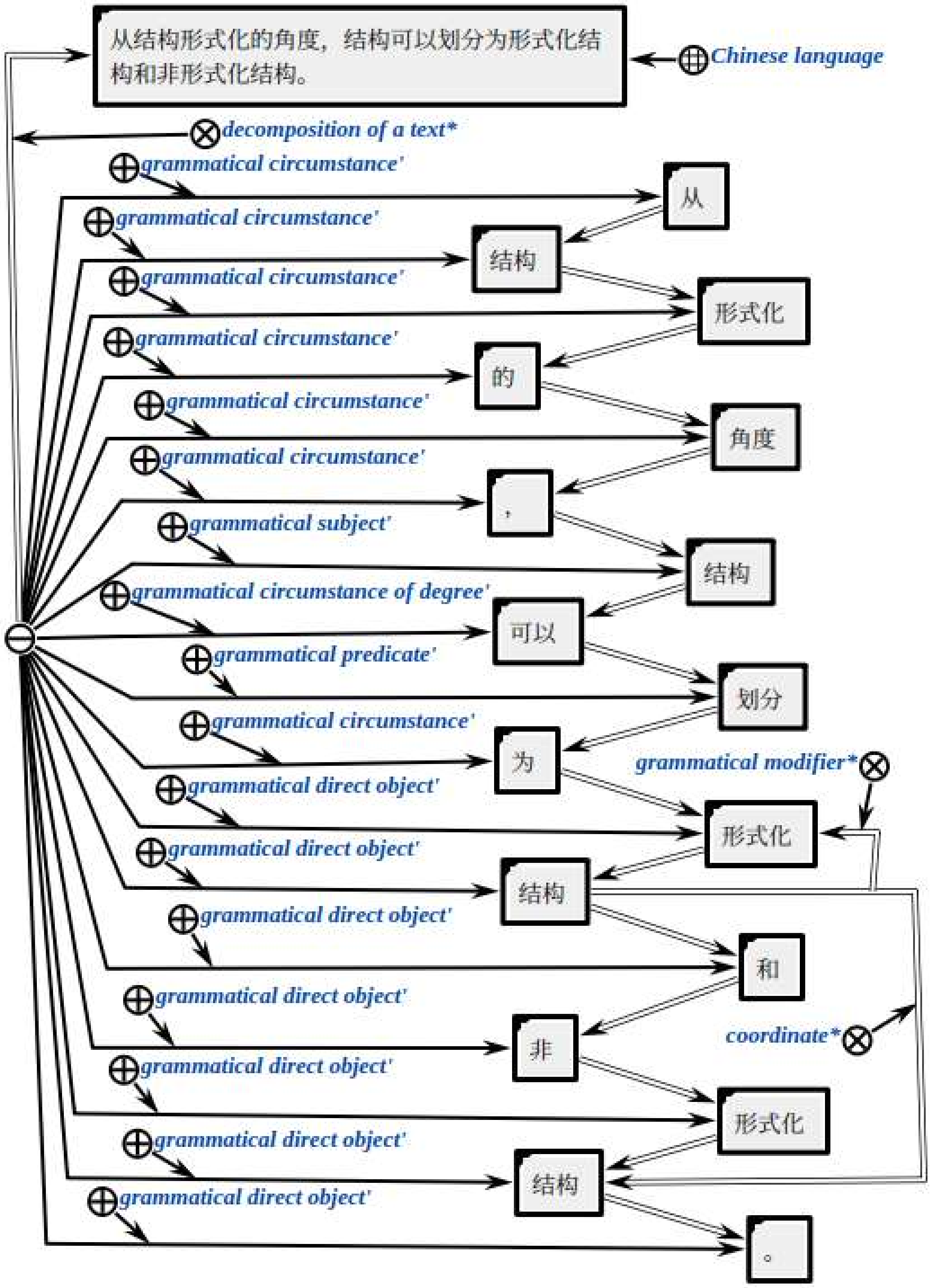


Figure 6: The result of lexical analysis for the input Chinese text



从结构形式化的角度，结构可以划分为形式化结构和非形式化结构。

Chinese language
decomposition of a text*
grammatical circumstance'
从
grammatical circumstance'
结构
grammatical circumstance'
形式化
grammatical circumstance'
的
grammatical circumstance'
角度
grammatical circumstance'
，
grammatical subject'
结构
grammatical circumstance of degree'
可以
grammatical predicate'
划分
grammatical circumstance'
为
grammatical modifier*
grammatical direct object'
形式化
grammatical direct object'
结构
grammatical direct object'
和
grammatical direct object'
非
coordinate*
grammatical direct object'
形式化
grammatical direct object'
结构
grammatical direct object'
。

Figure 7: The semantically equivalent fragment of the input Chinese text

named entities and relations, with the extracted results stored in the form of SC code. In fact, within OSTIS Technology, a relation is treated as a special entity that specifies a certain relation between pairs of independent named entities, which generally is represented respectively as the sc-node denoting the non-role relation or the sc-node denoting the role relation in the form of SCg-code.

In general, the pairs of independent named entities should appear in the analyzed syntactic semantic structure as noun phrases that belong to the nominative units. Afterwards the connections between these noun phrases and the input Chinese sentence, as well as the path linking the two noun phrases through other segmentation units, will reflect the corresponding relations between the pairs of named entities. More precisely, the noun phrases appearing in the analyzed syntactic semantic structure of input Chinese sentence are represented as identifiers of sc-nodes denoting the name of some named entities or some concepts stored in the knowledge base of intelligent help system for discrete mathematics.

In the Figure 8 shown the determination of named entities, the syntactic semantic relations between them and between named entities and the input sentence, that is, the determination of named entities , the syntactic semantic relations between them in the input Chinese sentence, as well as the syntactic semantic relations between named entities and the input Chinese sentence.

In the Figure 9 shown the resulted constructed fragment of the knowledge base from the input Chinese sentence in the intelligent help system for discrete mathematics without contradiction detection.

As can be seen from the Figure 9, in this case, a fragment of the knowledge base can be directly constructed without linking the named entities mentioned in the input source Chinese sentence with the corresponding exiting defined entities in the knowledge base of intelligent help system for discrete mathematics. In some cases, for a named entity in the knowledge base, there are different names in the natural language texts to describe this entity. In this case, it is necessary to perform contradiction elimination to relate different named entities (precisely identifiers of named entities) in natural language texts to the same named entities in the knowledge base of ostis-systems.

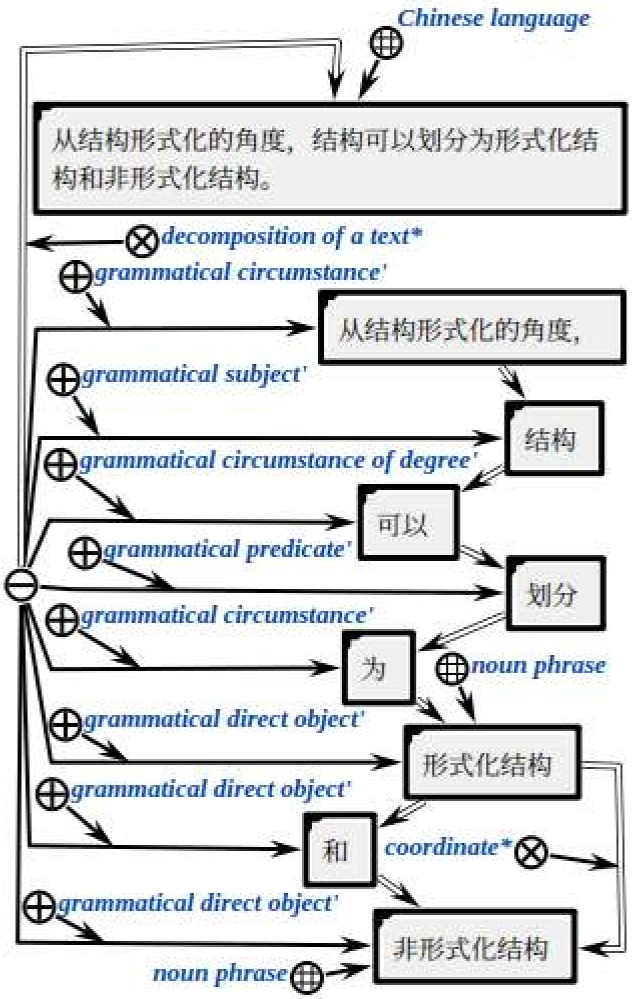

Chinese language
从结构形式化的角度，结构可以划分为形式化结构和非形式化结构。
decomposition of a text*
grammatical circumstance'
从结构形式化的角度，
grammatical subject'
结构
grammatical circumstance of degree'
可以
grammatical predicate'
划分
grammatical circumstance'
为
noun phrase
grammatical direct object'
形式化结构
grammatical direct object'
和
coordinate*
grammatical direct object'
非形式化结构
noun phrase

Figure 8: The determination of named entities and syntactic semantic relations in the input Chinese sentence

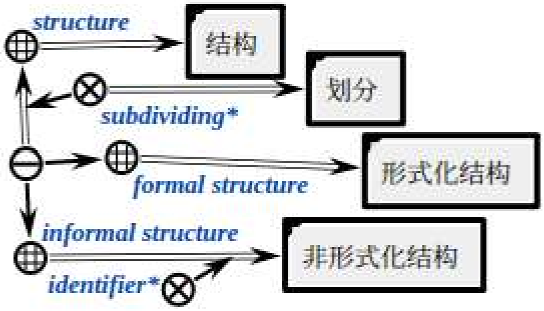


Figure 9: The constructed fragment of knowledge base from the input Chinese sentence

B. text generation from knowledge base

The current version of the component of Chinese texts generation from fragments of knowledge base for the intelligent help system for discrete mathematics is implemented in accordance with the corresponding general text generation architecture. In knowledge base on Chinese language processing, in addition to constructing linguistic knowledge (e.g., rules for text processing, extraction rules), linguistic knowledge for text generation (e.g., templates for text generation) can also be constructed in certain subject domains of knowledge base.

For fragments of knowledge base of ostis-system, the realization of natural language generation is roughly divided into two steps: rule-based symbolic generator converting fragments (sc-structure) of knowledge base into message triples; rule and template-based approached or statistical generator (when high quality aligned datasets is accessible.) translating message triples to resulted natural language texts. As mentioned above, unfortunately, the high quality aligned datasets is relatively difficult to access. The example about text generation is just to demonstrates the generation process of a simple narrative Chinese sentence using the rule template-based approach.

Due to the complexity and diversity of text generation tasks, in the example about text generation there are following several constrains:

- The input fragment of knowledge base is completed and has sense;
- The input fragment formally represented in the knowledge base for discrete mathematics;
- The output of Chinese language interface is a simple narrative Chinese sentence;
- The sc-nodes denoting concepts, named entities or relations within the input fragment of knowledge base have corresponding identifiers in Chinese language, which might be used in the resulted sentence.

Step 1 The Chinese language interface is provided with a given fragment (sc-structure) from knowledge base of intelligent help system for discrete mathematics represented in the form of the SC-code. For visual representation of given fragment of knowledge base, the fragment formally is represented in SCg (Figure 10).

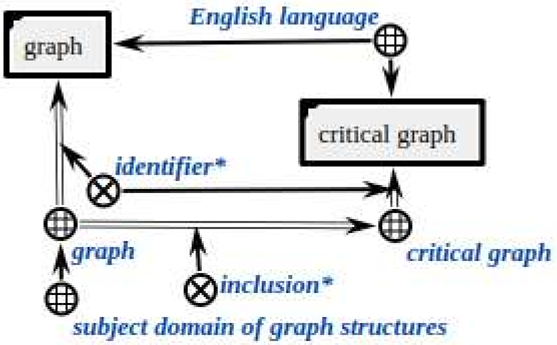


Figure 10: The fragment (sc-structure) of knowledge base for discrete mathematics

Step 2 The given fragment is divided into standard basic sc-constructions, afterwards from which the candidate sc-constructions are selected to be generated to the resulted Chinese texts. The candidate sc-construction (belong to standard basic sc-construction) is shown in SCg (Figure 11).

Step 3 The candidate sc-construction is transferred to the message triple, each sc-element of message triples is a file (an sc-node with content) corresponds to a certain lexical unit (e.g., in the Figure 12 shown the sc-node "I_ch_graph") in the Chinese language. Specification of lexical units that is stored in the knowledge base on Chinese language processing. As seen in the Figure 12, the example just consider one type of the set of message triples.

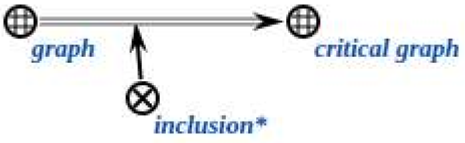


Figure 11: The determination of candidate sc-construction

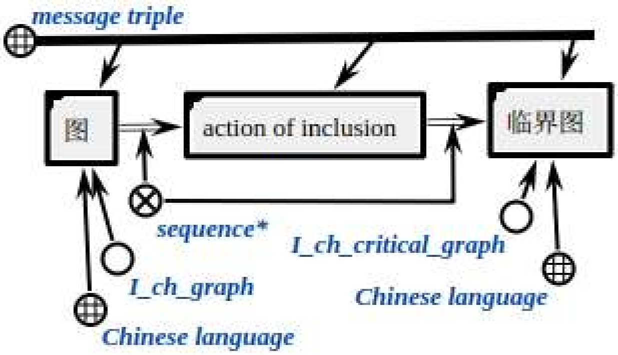


Figure 12: The message triple for candidate sc-construction

It is important to note that action of each message triple is the core that need to be replaced with corresponding text fragment (e.g., verb phrases or others) in order to generate fluent texts. Moreover, in general the subject and object of each message triple remain the same. Unless, in some cases of generating complex texts, they can be replaced by pronouns in the resulted texts.

Step 4 Based on the rule-template approach, for action of each message triple, a suitable rule or template constructed as logical ontologies in the knowledge base on Chinese language processing can be matched. For this example, the specific template shown in the Figure 14 will be applied.

Step 5 The agent fills the sc-links of corresponded sc-elements of that message triple, i.e. a certain sc-link needs to be filled with the result of a certain inflection form of a lexical unit. Finally the sc-links are concatenated to generate the resulted Chinese sentence according to valid ordering (Figure 13).

In the knowledge base there are different identifiers for each lexical unit in natural language. For Chinese language, there is not a certain inflection form for lexeme, therefore a lexical unit "graph" is expressed in resulted Chinese sentence is same as itself (Figure 13). Unlike European languages, in the resulted generated texts the lexical units in the sc-links require a specific inflection form (e.g., singular or plural and other inflection form) according to the syntax of a specific language. However

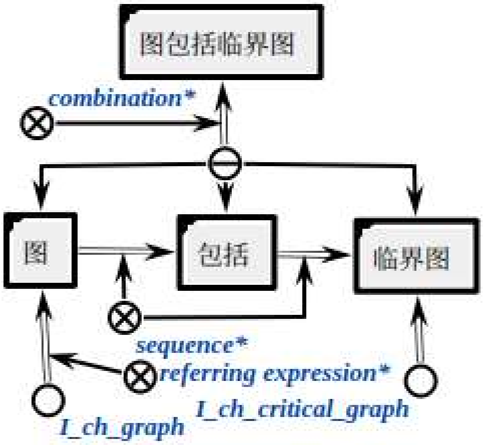


Figure 13: The resulted generated Chinese sentence corresponding to sc-construction

due to the features of Chinese language, the processing of this step is relative simpler. In this example we just consider the generation of the simple declaration sentence, the referring expression of the lexical unit means that the final form of this lexical unit in the resulted text. Based on the template, the referring expression to the lexical unit "graph" is served as the subject. The lexical unit "critical graph" is served as object in Chinese language. In some cases, the template is predefined with fixed phrase, it means that there are some sc-link in the template without corresponding lexical unit is the fixed part [41].

When these steps are implemented, a fragment sc-structure of knowledge base of Discrete Math domain can be transferred into Chinese sentence with specific sense. This natural language text is easier to access to ordinary end-users.

IX. Conclusion

This article has proposed an ontological approach based on the OSTIS Technology to develop a unified semantic model for natural language interface of knowledge-based intelligent system, which has ability of converting the natural language texts into the fragments of knowledge base and generating the natural language texts from the fragments of knowledge base. Within the framework of OSTIS Technology the development of semantic model for natural language interface mainly requires the development of sc-model of knowledge base of linguistics, which represented in the form of described subject domains and corresponding ontologies about linguistic knowledge, as well as sc-model of problem solver of natural language interface, which consists of sc-agents developed independently each other using various problem solving models to solve corresponding tasks in the natural language interface. The semantic model of natural language interface is universal and can be used for the implementation of various natural language interface of knowledge-based intelligent system in the specific subject domain. Within the developed semantic model of natural language interface, according to the features of the specific natural language, it becomes possible to develop knowledge base, within which includes linguistic knowledge in various level (lexical, syntactic, semantic and so on) about the specific natural language, and the corresponding problem solver on the specific natural language processing, in turn, to implement the specific natural language interface of knowledge-based intelligent system.

The developed semantic model of natural language interface using ontological approach mainly offers the following advantages:

- The component of converting the natural language texts into the fragments of knowledge base is applied to extract the knowledge structure represented in the form of SC-code from the natural language texts without predefined categories of entities and relations (i.e. from open domains);
- The component of generating the natural language texts from the fragments of knowledge base is applied to generate fluent, coherent, and multi-sentence natural language texts appropriating for end-users. For text generation, the semantic structures (fragments of knowledge base) that processed by this component is more complex than simple tabular or triples structure;
- The knowledge base of linguistics is constructed in the form of subject domains that are as independent of each other as possible and corresponding ontologies about linguistic knowledge. The hierarchical structure makes it possible to consider linguistic knowledge in various level (lexical, syntactic, semantic and so on) into a single knowledge base, as well as reduce development complexity of knowledge base and increase the development efficiency;
- The multi-agent model to develop the problem solver of each component in the natural language interface is possible to integrate different approaches to extend, modify the sc-agents to improve the performance of interface independently, without any change in other sc-agents;
- The modular and component approaches are considered to develop the natural language interface underlying a unified semantic basis. It makes sure the efficiency and semantic uniformity of each component development in the natural language interface of knowledge-based intelligent system.

We discussed the general structure of knowledge base of linguistics, which provides various types of linguistic knowledge to knowledge extraction and text generation. In addition to general declarative knowledge, within the constructed knowledge base includes knowledge for logical reasoning represented in logical ontologies

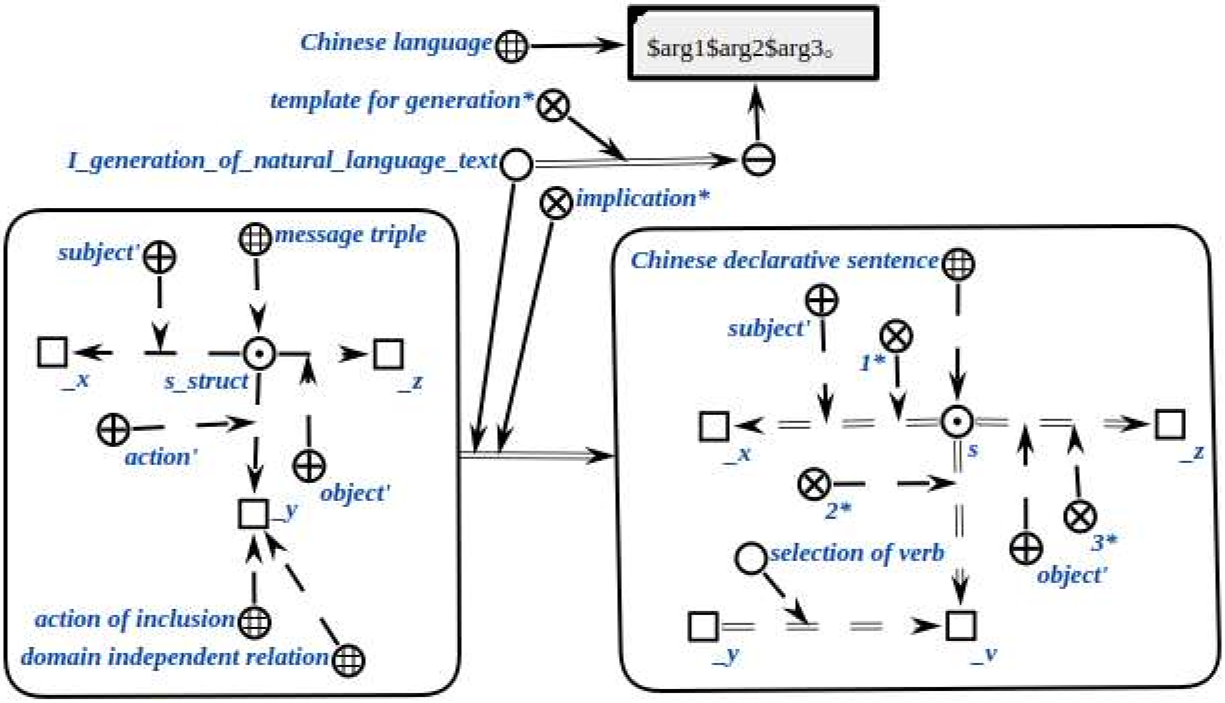


Figure 14: The template for generating natural language text

(e.g., the rules for knowledge extraction, the rules or templates for text generation). Relying on the sc-agents of each component to use constructed static linguistic knowledge to realize automatic knowledge extraction and text generation, it makes the intelligent system to obtain knowledge to help end-users, as well as the information of intelligent system easier accessible not only to computer programs, but also to end-users.

Finally the Chinese language interface of intelligent help system for discrete mathematics is implemented in the trial on the basis of the proposed semantic model of natural language interface to verify the practicality and generalization of the semantic model. In addition to Chinese language, it would be particularly interesting to explore the semantic model's possibility to support knowledge extraction and text generation for multiple language as long as the corresponding knowledge base on specific natural language processing (ontologies on specific natural language processing) and suitable sc-agents are developed.

References

[1] Liu Y. C.: Survey on Domain Knowledge Graph Research. Computer Systems Applications, 2020, vol. 26 No 06, pp. 1-12.

[2] Knowledge graph: the foundation for big data semantic link. Available at: http://www.cipsc.org.cn/kg2/. Date of access: 03.05.2022.

[3] Commonsense knowledge (artificial intelligence). Available at: https://en.wikipedia.org/wiki/Commonsense_knowledge_(artificial_intelligence)/. Date of access: 01.09.2022.

[4] Xu Z. L., Sheng Y. P., He L. R., Wang Y. F.: Review on Knowledge Graph Techniques. Journal of University of Electronic Science and Technology of China, 2016, vol. 45 No 04, pp. 589-606.

[5] Qian, L. W.: Ontological approach to Chinese text processing. Doklady BGUIR, 2020, vol. 18 No 06, pp. 49-56. (In Russian).

[6] Zhao H. X., Li L., Wu X. D., He J.: Knowledge Graph Oriented Information Extraction. Hans Journal of Data Mining, 2020, vol. 10 No 04, pp. 282-302.

[7] Li D. M., Zhang Y., Li D. Y., Lin D. Q.: Review of Entity Relation Extraction Methods. Journal of Computer Research and Development, 2020, vol. 57 No 07, pp. 1424-1448.

[8] Schutz A., Buitelaar P.: RelExt: A Tool for Relation Extraction from Text in Ontology Extension. International Semantic Web Conference, Springer, Berlin, Heidelberg, 2005, pp. 593 – 606.

[9] Banko M. J., Soderland S. Cafarella: Open Information Extraction from the Web. Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, 6-12 January 2007, pp. 2670-2676.

[10] Wu F., Weld D. S.: Open Information Extraction Using Wikipedia. Proceedings of Annual Meeting of the Association for Computational Linguistics, Uppsala, 11-16 July 2010, pp. 118-127.

[11] Fader A., Soderland S., Etzioni O.: Identifying Relations for Open Information Extraction. Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, John McIntyre Conference Centre, Edinburgh, 27-31 July 2011, pp. 1535-1545.

[12] Schmitzm M., Bai R., Soderiand S.: Open Language Learning for Information Extraction. Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju, Island, 12-14 July 2012, pp. 523-534.

[13] Tseng Y. H., Lee L. H.: Chinese Open Relation Extraction for Knowledge Acquisition. Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, Gothenburg, Sweden, 26-30 April 2014, pp. 12-16.

[14] Gatt A., Krahmer E.: Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. Journal of Artificial Intelligence Research, 2018, vol. 61, pp. 65-170.

[15] Theune M., Klabbers E., Pijper de: From data to speech: a general approach. Natural Language Engineering, 2001, vol. 07 No 01, pp. 47-86.

[16] Yao T. F., Zhang D. M., Wang Q.: System Demonstration

Multilingual Weather Forecast Generation System. In Natural Language Generation, Niagara-on-the-Lake, Ontario, Canada, 1998, pp. 296-299.

[17] Reiter E. Sripada S., Hunter J. R.: Choosing words in computer-generated weather forecasts. Artificial Intelligence, 2005, vol. 167 No 01-02, pp. 137-169.

[18] Plachouras V., Smiley C., Bretz H.: Interacting with financial data using natural language. In Proc. SIGIR'16, Italy, Pisa, 2016, pp. 1121-1124.

[19] Androutsopoulos I., Lampouras G., Galanis D.: Generating Natural Language Descriptions from OWL Ontologies: the NaturalOWL System. Journal of Artificial Intelligence Research, 2013, vol. 48 No 01, pp. 671-715.

[20] Xia Z. T., Qu W. G., Gu Y. H., Zhou J. S., Li B.: Review of Entity Relation Extraction based on deep learning. In Proceedings of the 19th Chinese National Conference on Computational Linguistics, Haikou, China, Chinese Information Processing Society of China, 2020, pp. 349–362.

[21] Zhuang C. Z., Jin X. L., Zhu W. J., Liu J. W., Bai L, Cheng X. Q.: Deep Learning Based Relation Extraction: A Survey. Journal of Chinese Information Processing, 2019, vol. 33 No 12, pp. 1-18.

[22] Li J. Y., Tang T. Y., Zhao W. X.: Pre-trained Language Models for Text Generation: A Survey. Thirtieth International Joint Conference on Artificial Intelligence IJCAI-21, Montreal-themed virtual reality, 19th -26th August 2021, pp. 4492–4499.

[23] Claire G., Anastasia S., Shashi N.: The WebNLG Challenge: Generating Text from RDF Data. In Proceedings of the 10th International Conference on Natural Language Generation, Santiago de Compostela, Spain, 2017, pp. 124–133.

[24] Auer S., Bizer C., Kobilarov G.: Dbpedia: A nucleus for a web of open data. The Semantic Web, Springer, Berlin, Heidelberg, 2007, pp. 722-735.

[25] Xu B., Xu Y., Liang J.: CN-DBpedia: A never-ending Chinese knowledge extraction system. International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, Cham, 2017, pp. 428-438.

[26] Niu X., Sun X. Wang H.: Zhishi.me – weaving Chinese linking open data. International Semantic Web Conference, Springer, Berlin, Heidelberg, 2011, pp. 205-220.

[27] Chinese General Encyclopedia Knowledge Graph (CN-DBpedia). Available at: http://www.openkg.cn/dataset/cndbpedia/. Date of access: 10.09.2022.

[28] Chinese Encyclopedia Knowledge Graph (Zhishi.me). Available at: http://openkg.cn/dataset/zhishi-me-dump/. Date of access: 10.09.2022.

[29] Amit M., Yoav G., Ido D.: Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), USA, Minneapolis, 2019, pp. 2267-2277.

[30] Golenkov V.V. Gulyakina N.A.: Proekt otkrytoi semanticheskoi tekhnologii komponentnogo proektirovaniya intellektual'nykh sistem. Chast' 1 Printsipy sozdaniya [Project of open semantic technology of component designing of intelligent systems. Part 1 Principles of creation]. Ontologiya proektirovaniya [Ontology of designing], 2014, No 1, pp. 42-64. (in Russian).

[31] Davydenko, I. T.: Ontology-Based Knowledge Base Design, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Belarus Minsk, 2017, pp. 57-72.

[32] Davydenko, I. T.: Semantic Models, Method and Tools of Knowledge Bases Coordinated Development Based on Reusable Components, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Belarus Minsk, 2018, pp. 99–118.

[33] Shunkevich D.V.: Ontology-Based Knowledge Base Design, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Belarus Minsk, 2017, pp. 73-94.

[34] Shunkevich D.V.: Ontological approach to the development of hybrid problem solvers for intelligent computer systems, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Belarus Minsk, 2021, pp. 63–74.

[35] Boriskin A. S., Sadouski M. E., Koronchik D. N., Zhukau I. I., Khusainov A. F.: Ontology-Based Design of Intelligent Systems User Interface, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Belarus Minsk, 2017, pp. 95–106.

[36] Sadouski M. E.: Ontological approach to the building of semantic models of user interfaces, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Belarus Minsk, 2021, pp. 105–116.

[37] Shunkevich D.V.: Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Belarus Minsk, 2018, pp. 119–132.

[38] Qian L. W., Sadouski M. E., Li W. Z.: Ontological Approach for Chinese Language Interface Design, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Minsk, 2020, pp. 146–160.

[39] Hardzei ., Svyatoshchik ., Bobyor L., Nikiforov S.: Processing and understanding of the natural language by an intelligent system, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Belarus Minsk, 2021, pp. 123–140.

[40] Qian L. W., Li W. Z.: Ontological Approach for Generating Natural Language Texts from Knowledge Base, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Belarus Minsk, 2021, pp. 159–168.

[41] Wang H., Zhan W. D., Yu S. W.: Structure and Application of The Semantic Knowledge base of Modern Chinese. Applied Linguistics, 2006, No 01, pp. 134-141.

[42] Contemporary Chinese language word segmentation specification for information processing. Available at: http://std.samr.gov.cn/gb/search/gbDetailed?id=71F772D78FC6D3A7E05397BE0A0AB82A/. Date of access: 01.07.2022.

Онтологический подход к разработке естественно-языкового интерфейса

Цянь Лунвэй, Ли Вэньцзу

В статье рассматриваются существующие методы к разработке двух основных компонентов в естественно-языковом интерфейсе интеллектуальнных систем, т.е. преобразование текстов естественного языка во фрагменты базы знаний и генерация текстов естественного языка из фрагментов базы знаний. Был проведен анализ проблем, возникающих при преобразовании текстов естественного языка во фрагменты базы знаний и генерации текстов естественного языка из фрагментов базы знаний в настоящее время.

На основании различных рассмотренных методов был предложен онтологический подход к разработке естественно-языкового интерфейса, который позволяет интегрировать разные типы лингвистических знаний и методов в единой семантичесской модели для анализа текстов естественного языка и генерации текстов есттетсвенного языка. Этапы реализации подхода были созданы лингвистические онтологии и решатели для анализа и генерации текстов естественного языка. Более того, в качестве китайского языка, функции каждых этапов анализа текстов и генерации текстов, а также назначение лингвистических онтологий в процессе генерации проиллюстрированы, чтобы проверять практичность модели.

Received 14.11.2022

Audio interface of next-generation intelligent computer systems

Vadim Zahariev, Kuanysh Zhaksylyk, Denis Likhachov, Nick Petrovsky, Maxim Vashkevich, Elias Azarov
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
{zahariev, likhachov, nick.petrovsky, vashkevich, azarov}@bsuir.by, kuanysh.zhk@gmail.com

***Abstract*—The article is dedicated to the issues of creating audio and voice interfaces for next-generation intelligent computer systems. It is proposed to use an approach based on ontological design and formalization of a concepts system from the subject domain of audio interfaces using the OSTIS Technology. The main ideas underlying this approach, as well as their features distinguishing them from the generally accepted ones, are outlined. It is shown that in the future, the usage of this approach can provide the properties of unification, semantic compatibility, and interoperability in the development of audio and voice user interfaces, which ultimately will significantly reduce costs when creating next-generation intelligent computer systems for solving complex problems.**

***Keywords*—audio interface and voice interface of intelligent computer systems; semantically compatible components; speech processing and digital signal processing**

I. Introduction

Spoken language is one of the most natural and effective forms of information exchange between humans. This fact explains the significant interest of researchers in the development and application of voice interfaces for human-machine interaction as part of modern communication, multimedia, and intelligent systems [1], [2].

A more comprehensive form of interaction with the user and the environment through the analysis and synthesis of acoustic signals is an audio interface. This type of interface, which acts as a maternal in relation to voice ones, can be briefly defined as a hardware-software complex that analyzes and synthesizes signals in the entire available spectrum of parameters of acoustic information carriers, for example, to solve the problems of analyzing the situation and events occurring in the acoustic environment of the system, synthesizing non-speech signals (technogenic and natural sounds, warning signals, music, etc.) [3].

The following main tendencies in the development of this direction indicate the relevance of the direction of developing audio and voice interfaces:

- economic indicators and forecasts for the development of the speech technologies market, the current average annual growth rate of which, according to experts, is about 22%, and the total volume will be equal to 59.6 billion US dollars by 2030 [4];
- the appearance of a wide range of products based on the voice interface, which have gained widespread. First of all, these are personal voice assistants, such as Alexa (Amazon), Siri (Apple), Cortana (Microsoft), Alice (Yandex) [5]–[7];
- interest from the scientific community, expressed in the growth of publications in this field of research by 15% over the past 5 years [8].

It should be noted that the basic mass of scientific publications in this direction is dedicated to the development of basic technologies that are components of the voice interface, such as text-to-speech synthesis, as well as speech-to-text transformation [9]–[11]. Recent achievements in these fields are associated with the rapid development of neural network models and computing tools. They made it possible to bring the qualitative characteristics of the usage of speech technologies to a commercial level [12], [13].

II. Problem statement

Most of the existing systems, as a rule, are designed to solve a certain range of problems and are hardly compatible with each other. This fact is especially acute when designing complex systems like intelligent personal dialog assistants (Figure 1), which require using a variety of different types of processed information and different problem-solving models. Such systems, in addition to standard modules for recognition (ASR, automatic speech recognition) and synthesis (TTS, text-to-speech), at the audio interface level, should also contain models that determine the presence/absence of speech in the audio signal in a complex acoustic environment, classify environmental sounds, recognize a speaker, etc. In addition, elements of the voice interface must be compatible with higher-level modules for processing natural language information, such as modules for speech understanding (SLU, spoken language understanding) and generation (SLG, spoken language generation), dialog control (DM, dialog manager) [14].

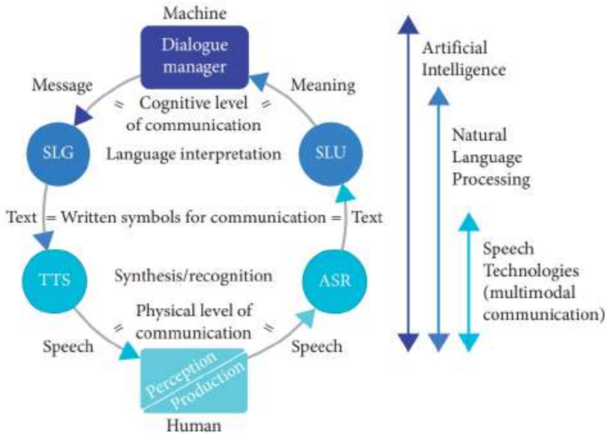

Machine
Dialogue manager
Message
Meaning
Cognitive level of communication
SLG
Language interpretation
SLU
Text
Written symbols for communication
Text
TTS
Synthesis/recognition
ASR
Physical level of communication
Speech
Speech
Perception
Production
Human
Artificial Intelligence
Natural Language Processing
Speech Technologies (multimodal communication)

Figure 1. Components of a human-machine speech dialog system [14].

All this requires the development of approaches based not only on machine learning methods and signal processing but also on natural language processing, symbolic methods of artificial intelligence, ontological design, and formalization of the subject domain of the audio interface. This will allow the creation of systems that have a full range of knowledge in a formalized form about the types of problems, that they must solve, and the methods available for solving them.

A necessary condition for the creation of such next-generation systems with improved characteristics in terms of interoperability and flexibility is also the fact that these systems must be built on the basis of a basic technology that allows such a unity of the form of information representation at all its levels.

The combination of these factors leads to the need to create next-generation intelligent computer systems that will include audio and voice interface modules based on the principles of interoperability and semantic compatibility to solve complex problems.

III. Suggested approach

To achieve this purpose, it would be advisable to use an approach based on the principles underlying the "Standard of the Open Technology for the Ontological Design, Production, and Operation of Semantically Compatible Hybrid Intelligent Computer Systems", or briefly the "Standard of the OSTIS Technology" [15].

The essence of the approach is to consider the process of designing an audio interface as an interface subsystem within the general process of developing an intelligent computer system (ICS) and building its formal logical-semantic model.

To create such a model of next-generation intelligent computer systems, it is necessary:

- to decompose an information computer system into components. The quality of the decomposition is determined by the simplicity of the subsequent synthesis of the general formal model from the formal models of the selected components;
- to carry out the convergence of selected components in order to build compatible (easily integrated) formal models of these components;
- to perform the integration of the built formal models of the selected components and obtain a common formal model.

The general methodological principles that are the basis for the transition to next-generation ICS are:

- convergence and unification of ICS and their components;
- structural-system simplification of ICS ("Occam's Razor" principle);
- orientation to universal ICS;
- synthesis of ICS from compatible components;
- orientation towards the creation of synergetic ICS.

The following important features of the proposed approach follow from the general methodological principles, which must be taken into account in order to achieve the purpose:

- semantic knowledge representation;
- agent-oriented basic model for processing knowledge bases that have a semantic representation (insertional programming in a semantic space);
- semantic structuring of knowledge bases in the form of a hierarchical system of subject domains and corresponding ontologies that specify these subject domains;
- at the same time, a next-generation ICS interface is interpreted as a specialized intelligent information system focused on solving interface problems of the corresponding individual ICS and deeply integrated (embedded) into this ICS.

As a technological basis for the implementation of the proposed approach, the OSTIS Technology [16] will be used. Systems built on the basis of the OSTIS Technology are called ostis-systems, respectively, the audio interface subsystem will be built as a reusable component, which in the future will be built into various ostis-systems, if necessary. As a formal basis for encoding various information in the knowledge base, an SC-code [16] is used, the texts of which (sc-texts) are written in the form of semantic networks with a basic set-theoretic interpretation. The elements of such networks are called sc-elements (sc-nodes, sc-arcs). The focus of this work on the OSTIS Technology is conditioned by its following main advantages:

- within this technology, unified means of representing various types of knowledge, including meta-knowledge, are proposed, which make it possible to describe all the information necessary for analysis in one knowledge base in a unified format [17];
- the formalism used within the technology allows specifying in the knowledge base not only concepts

but also any files external from the point of view of the knowledge base (for example, fragments of a speech signal), including the syntactic structure of such files;
- the approach proposed within the technology for representing various types of knowledge [17] and their processing models [18] ensures the modifiability of ostis-systems, i.e. allows easily expanding the functionality of the system by introducing new types of knowledge (new concepts systems) and new models of knowledge processing.

In this work, unlike the previous ones, which deal with the issues of semantic analysis of voice messages based on a formalized context [19] and the creation of dialog assistants based on a mental lexicon model [20], [21] or a multimodal system based on a neurosymbolic approach [22], OSTIS technology is used to directly build an ontology of the audio interface subsystem.

Since the next-generation ICS audio interface must have an architecture that corresponds to the general rules for building ostis-systems, the following main parts of it can be distinguished and formalized:

Audio interface of next-generation intelligent computer systems
⇒ *reduction**:
  [ICS audio interface]
⇒ *generalized decomposition**:
  {• *knowledge base of the subsystem of the next-generation ICS audio interface*
  • *problem solver of the subsystem of the next-generation ICS audio interface*
  • *interface for interacting with other interface subsystems of the ostis-system*
  }

Thus, it should be noted that the process of developing an audio interface for next-generation ICS implies, first of all, the creation of semantically structured knowledge bases in the form of a hierarchical system of subject domains and corresponding ontologies that specify these subject domains. Therefore, the first step to achieve this purpose is the phase of identifying and formalizing the entities of the audio and voice interface in order to immerse this information in the knowledge base of an intelligent computer system.

From our point of view, it is possible to decompose the subject domains and ontologies included in the knowledge base of the audio interface into the following main directions:

Subject domain and ontology of the audio interface of next-generation intelligent computer systems
⇐ *decomposition**:
  {
  • *Subject domain and ontology of audio interface problems*
  • *Subject domain and ontology of signal parametric representation models*
  • *Subject domain and ontology of signal parameter classification models*
  }

As it is shown, the functional approach to the decomposition of subject domains is put at the head of the ontology, which is quite natural because it corresponds to the nature of the problems implemented by the audio interface.

The principles, represented above, together allow for the convergence and integration of components both at the level of the audio interface subsystem and at the level of the entire ICS as a whole, which, in turn, allows “transfering” an intelligent information system into a class of hybrid, interoperable, and synergistic systems.

Next, we proceed directly to the consideration of specific subject domains and the building of an ontology of the audio interface.

IV. Subject domain and ontology of audio interface problems

The first step towards building the knowledge base of the subsystem of the next-generation ICS audio interface is the formalization of the top-level ontology. This ontology is proposed to be based on a formalized representation of the main entities of the subject domain and their properties, as well as functional problems that the audio and voice interface are designed to solve.

The main entities, which require formalization and immersion into the knowledge base, include the set of concepts represented below. One of the key concepts requiring formalization is the basic definition of the signal itself, as well as the main types of signals, depending on their nature, which are of the greatest interest in the field of audio interfaces. To make the description process by means of the OSTIS Technology clearer, before proceeding directly to it, we will give examples of the main entities and concepts that require formalization and immersion into the knowledge base:

- signal;
- acoustic signal;
- audio signal;
- speech signal.

Depending on the way of mathematical description of the processed signal in the ostis-system, the following classes can be allocated:

- analog signal;
- discrete signal;
- digital signal;
- periodic signal;
- aperiodic signal;
- harmonic signal;

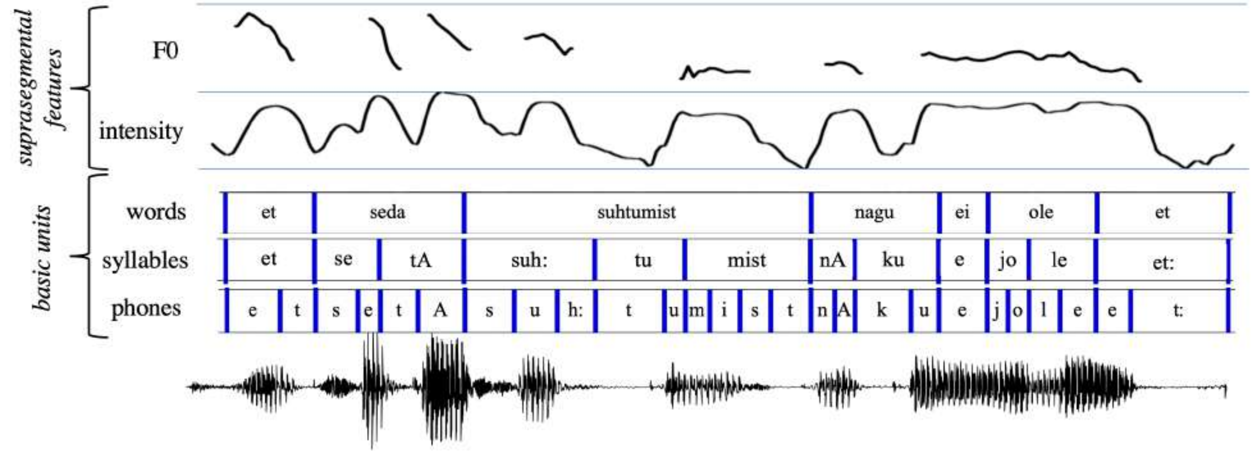


Figure 2. Segmental and suprasegmental features of the speech signal [23]

- tone signal;
- noise signal;
- pulse signal.

To successfully immerse the necessary knowledge for the operation of the audio interface, it is also necessary to formalize the basic concepts associated with the features of the signal itself, according to the following main attributes:

- signal amplitude;
- signal frequency;
- signal phase;
- signal intensity;
- signal duration;
- signal power/energy;
- signal oscillogram;
- signal spectrogram;
- signal discretization interval;
- signal quantization degree.

The key concepts of the subject domain lying in the semantic neighborhood of the subspace of the functional purpose for audio interfaces and audio signal processing are the following ones:

- audio signal analysis;
- audio signal synthesis;
- audio signal encoding;
- audio signal denoising;
- audio signal classification;
- environmental sound classification;
- acoustic scenes and events classification;
- anomalous sound detection;
- sound source localization.

The basic concepts of audio and voice interface are also closely related to its main features, which can be divided into the following main groups of concepts:

- speech signal features;
- linguistic features of the speech signal;
- paralinguistic features of the speech signal;
- extralinguistic features of the speech signal;
- segmental features of the speech signal;
- suprasegmental features of the speech signal;
- speech signal volume;
- speech signal timbre;
- speech signal rate;
- frequency of the main signal tone;
- phonemic composition of the speech signal.

The main concepts of the subject domain lying in the semantic neighborhood of the functional purpose of the speech and audio signal processing are the following ones:

- speech signal analysis;
- speech signal synthesis;
- speech recognition;
- emotional speech recognition;
- text-to-speech synthesis;
- emotional text-to-speech synthesis;
- sing synthesis;
- voice activity detection;
- key words spotting;
- wake up word detection;
- speech diarization;
- speaker recognition;
- speaker classification;
- speaker verification.

It should be noted that the above concepts are often interconnected in a complex and non-trivial way in the process of transition from information sources to direct physical parameters. Such a complex signal structure can be represented as a diagram of its information structure (Figure 3). This fact requires next-generation ICS to formalize concepts, so that the system can automatically interpret the interconnections between these features, when working with audio and speech signals, and, as a result, supply a response to the user, explaining on the

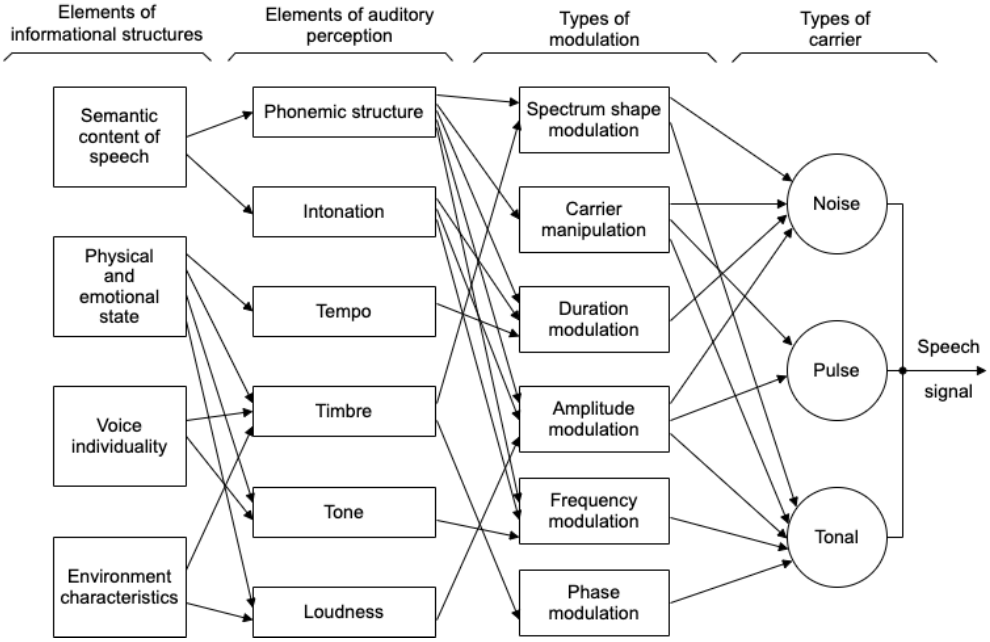


Figure 3. Speech signal features and their interconnections [24]

basis of what features the system came to a particular conclusion.

Since for the building of next-generation ICS, the focus is precisely on the problems associated with the processing of speech signals, the solution of which is necessary first of all to build a voice interface, we will try to focus on the formalization features of this subject domain.

The basis of the SC-model of the knowledge base is a hierarchical system of subject domains and their corresponding ontologies. The top level of the hierarchy for the part of the knowledge base related directly to the audio and voice interfaces is shown below.

Here is a formalized representation of some of the above concepts:

signal

⇒ *definition**:
 [physical process that carries a message (information) about some event, the state of the considered object, or issues the control, alerts commands, etc.]

⊃ *acoustic signal*
 ⇒ *definition**:
  [signal representing the propagation of elastic waves in a gaseous, liquid, or solid medium]

⊃ *audio signal*
 := [sound signal]
 ⇒ *definition**:
  [acoustic signal whose parameters are within the range of values accessible to human senses]
 ⊃ *acoustic signal*
 ⇒ *note**:
  [The frequency range of the audio signal is between 20 and 20,000 Hz.]

⊃ *speech signal*
 ⇒ *definition**:
  [audio signal generated by the passage of air flows through the human vocal tract. As a result of various acoustic transformations, the formation of various speech sounds occurs]
 ∋ *verbal speech*
 ∋ *speech production*
 ⇒ *note**:
  [The mechanism of human speech production is an acoustic tube with dynamically changing cross-sectional parameters, excited either by a quasi-periodic sequence of impulses generated by the vocal cords or by a turbulent flow of air pushed through constrictions in different parts of the vocal tract.]

Formalization at the level of a top-level ontology will be represented only for the basic concepts of the subject domain of signal models. The varieties of models of mathematical representation will be discussed in the corresponding subsection below.

Depending on the model for representing the signal itself in the ostis-system, the following descriptions of the main types of signals can also be defined, the usage of which is justified by the nature of the analyzed signal, as well as the analysis of the problem to be solved:

signal model
⇐ *combination**:
{• *analog signal*
⇒ *definition**:
[signal whose parameters can be measured at any time]
⇒ *definition**:
[signal where each of the represented parameters is described by a function of time and a continuous set of possible values]
• *discrete signal*
⇒ *definition**:
[signal for which at least one of the represented parameters is described by a finite set of possible values]
⇐ *combination**:
{• *discrete in time*
• *discrete in amplitude*
}
• *digital signal*
⇒ *definition**:
[signal where each of the representing parameters is described by a discrete time function and a finite set of possible ones]
⇒ *subdividing**:
{• *signal discrete in time*
• *signal quantized (discrete) in amplitude*
}
• *periodic signal*
• *aperiodic signal*
• *tone signal*
• *harmonic signal*
• *pulse signal*
• *noise signal*
}

It should be noted that due to restrictions on the size of the material for the features of the audio signal, we will give only a hierarchy of their general interrelations, since the semantics of these concepts is quite typical for other fields of technical sciences and does not require detailed examples and explanations.

audio signal features
⇐ *combination**:
{• *signal amplitude*
• *signal frequency*
• *signal phase*
• *signal intensity*
• *signal duration*
• *signal power*
• *signal spectrum*
• *signal oscillogram*
⇒ *definition**:
[function that fixes the dependence of changes in signal features (first of all, amplitude) in time]
• *signal spectrogram*
⇒ *definition**:
[function that fixes the dependence of the power spectral density of an audio signal in time]
• *signal discretization interval*
⇒ *definition**:
[value of the frequency with which the signal was discretized over time during the analog-digital conversion]
⇐ *typical values**:
{• *8000 Hz*
• *16000 Hz*
• *22050 Hz*
• *44100 Hz*
• *48000 Hz*
}
• *signal quantization degree*
⇒ *definition**:
[permissible number of discrete signal levels expressed as a degree of two and used in the process of quantization of the signal by level in the process of analog-digital conversion]
⇐ *typical values**:
{• *8 bits*
• *10 bits*
• *12 bits*
• *16 bits*
• *24 bits*
}
}

The description of the subject domain of signal models will be considered in more detail in the next subsection of the article, so we do not consider it necessary to demonstrate it here. We formalize the ontology of the main features of a speech signal in the following form [25]:

speech signal features
⇐ *combination**:
{• *communicative features*

⇒ *note**:
[encode the meaning of the transmitted message and depend on the sender’s intentions]
• *informative features*
⇒ *note**:
[encode additional information of the transmitted message and do not depend on the sender’s intentions]
• *informative features*
⇒ *note**:
[encode additional information of the transmitted message and do not depend on the sender’s intentions]
• *segmental features of the speech signal*
⇒ *note**:
[carry information about the current state of the source for the duration of one or more phonetic units]
• *suprasegmental features of the speech signal)*
⇒ *note**:
[carry information about the state of the source and the transitions between them throughout the entire utterance]
}
⊃ *linguistic features of the speech signal*
∋ *verbal communication means*
∋ *communicative features*
⇒ *definition**:
[features carrying information by means of usage of the human language coding system]
⇒ *note**:
[linguistic features include both phonological (segmental and supersegmental) and grammatical code (morphology and syntax). Linguistic communication informs the receiver of the sender’s intentions through explicit verbal forms]
⊃ *paralinguistic features of the speech signal*
∋ *non-verbal communication means*
∋ *communicative features*
⇒ *definition**:
[features carrying information through additional means of communication, not directly related to the language]
⇒ *note**:
[transmit information about the attitude towards the subject of conversation, feelings, or emotional state of the speaker]
⇐ *combination**:
{• *speech intonation*
⇐ *combination**:
{• *frequency of the main signal tone* F_0
• *changing the frequency of the main signal tone* ΔF_0
}
• *speech volume*
⇐ *combination**:
{• *signal amplitude*
• *signal intensity*
}
• *speech tempo*
• *pause duration*
}
⊃ *extralinguistic features of the speech signal*
∋ *informative features*
⇒ *definition**:
[features that do not directly encode the meaning of the message but contain additional information about the sender and the conditions of communication]
⇒ *note**:
[transmit information about the attitude towards the subject of conversation, feelings, or emotional state of the speaker]
⇐ *combination**:
{• *narrator voice features*
⇐ *combination**:
{• *pitch*
• *timbre*
• *volume*
}
• *acoustic environment*
}

In Figure 4, the result of formalization of the subject domain and the ontology of typical problems of audio and voice interfaces for next-generation ICS by means of the SCg language is represented.

It should be noted that in this article we will give examples and focus only on the formalization of some of the concepts outlined above. The second important note is that the represented set of concepts is by no means exhaustive, since the main results of the work on the formalization of this subject domain will be discussed within more voluminous works such as a monography and the next versions of the OSTIS standard.

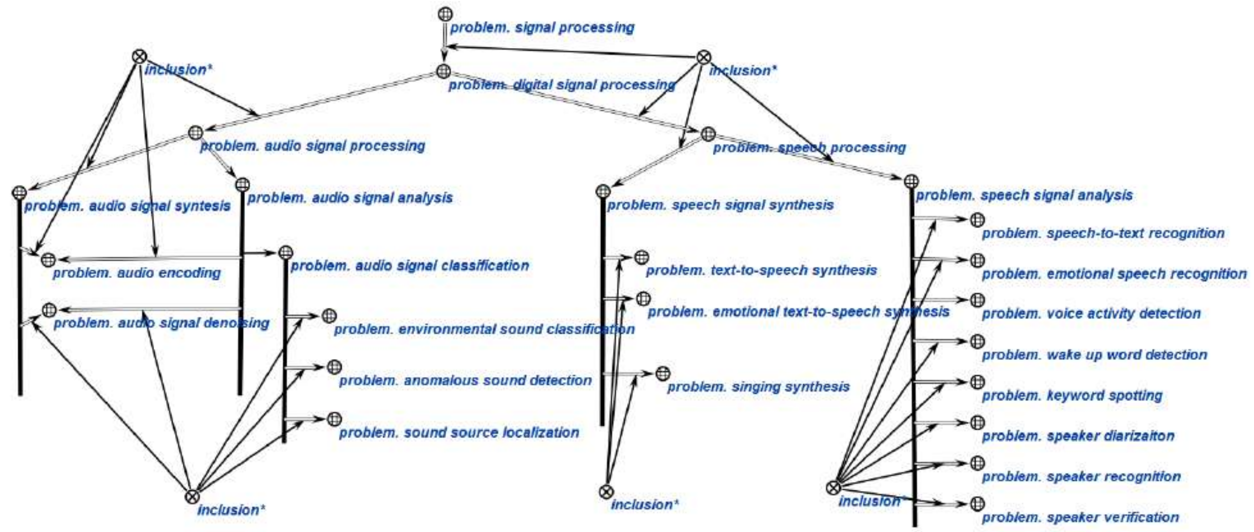

problem. signal processing
inclusion*
inclusion*
problem. digital signal processing
problem. audio signal processing
problem. speech processing
problem. audio signal syntesis
problem. audio signal analysis
problem. speech signal synthesis
problem. speech signal analysis
problem. audio encoding
problem. audio signal classification
problem. text-to-speech synthesis
problem. speech-to-text recognition
problem. emotional speech recognition
problem. audio signal denoising
problem. environmental sound classification
problem. emotional text-to-speech synthesis
problem. voice activity detection
problem. wake up word detection
problem. anomalous sound detection
problem. singing synthesis
problem. keyword spotting
problem. sound source localization
problem. speaker diarizaiton
problem. speaker recognition
inclusion*
inclusion*
inclusion*
problem. speaker verification

Figure 4. The top-level ontology fragment of problems of audio and voice interfaces for next-generation intelligent computer systems

V. Subject domain and ontology of signal parametric representation models

All of the above problems are interrelated, since they refer to the same object of research – the speech signal. The solution of each of them directly or indirectly depends on the effectiveness of speech modeling as a complex phenomenon in various aspects: parametric representation of the speech signal and allocation of its properties, modeling the process of phonation, perception, and interpretation of the contents of a speech message (including phonetic, semantic, emotional ones). This makes the creation of universal methods for processing speech signals a promising scientific direction. In the context of the above problems, speech modeling can be conditionally divided into three levels:

- general signal modeling using samples in the time or frequency domain;
- modeling of signal features that are specific to speech and related to the phonation process (such as frequency of the main tone, excitation sequence, and amplitude spectrum envelope);
- modeling of high-level speech features (voice, accent, expression, phonetic and semantic contents of a speech message). Each next level is based on the previous one and implies the usage of special methods of parametric description.

The first two levels include models widely known in digital processing of speech signals based on linear prediction (LP), cepstral coefficients, and sinusoidal parameters.

Among the approaches using the sinusoidal description of the signal, currently, the most promising are mixed (hybrid) models, which take into account the possibility of different modes of phonation with the participation of the vocal cords (voiced speech) and without the participation of the vocal cords (unvoiced speech), moreover, each of these two modes is described by the corresponding model (Figure 5).

Voiced speech is considered as a quasi-periodic (deterministic) signal, while unvoiced speech is considered as a non-periodic (stochastic) signal. The most famous among existing models is the harmonic + noise model, which is used to solve such complex problems as the creation of voice interfaces, speech recognition, text-to-speech synthesis, voice conversion, noise reduction, increasing the intelligibility and subjective quality of speech signals, accent correction, and so on. Its advantage is the theoretical possibility of modeling vocalized sounds in the form of continuous functions with varying parameters, which makes it possible to obtain an effective description of the phonation process and avoid the overlap of adjacent fragments, phase breaks in speech synthesis. The disadvantage of the model is the high complexity of the analysis and synthesis algorithms due to the non-stationarity of the speech signal [26]–[29].

Since voiced speech consists of quasi-periodic components with varying parameters, it is necessary to use digital filters with variable features for analysis: their bandwidth must change in accordance with the contour of the frequency of the main tone. This requires the usage of special time-frequency transformations that allow estimation of periodic components with strong frequency modulation such as Fan-Chirp and harmonic transformations. The accuracy of parameter estimation is directly related to the accuracy of estimating the contour of the frequency of the main tone, so the usage of a reliable and accurate estimation method is a necessary condition for the successful usage of this model [30]–[32].

Another difficult problem is the automatic separation of

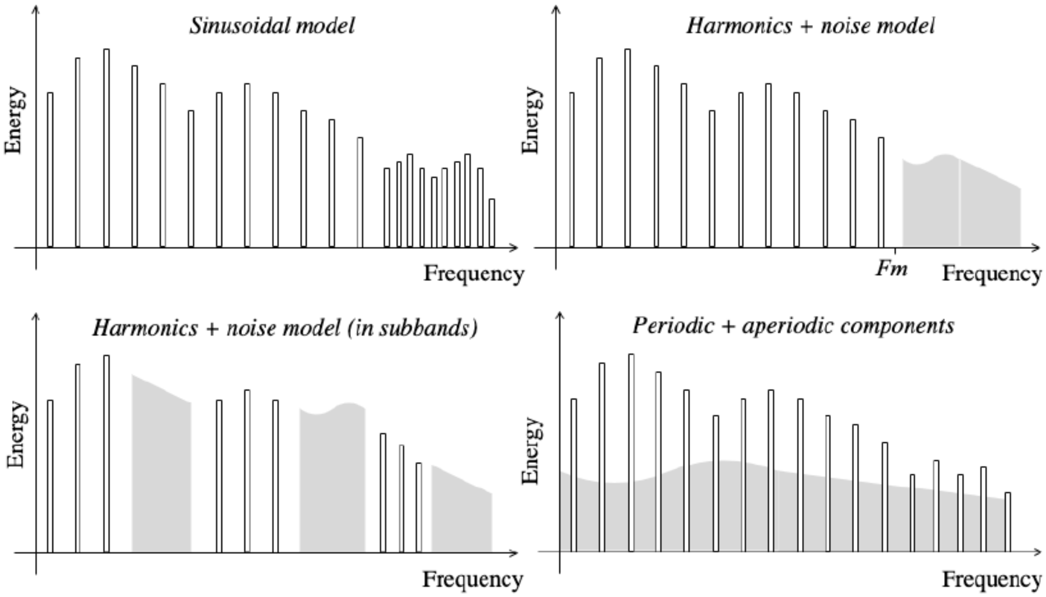


Figure 5. Classification of common speech signal models [26]–[32]

the signal into deterministic and stochastic components, for which special periodicity detectors are used.

Modeling a speech signal based on LP is a classic approach that has been used in digital speech processing for quite a long time. The main advantage of the model is the separate description of the signal in the form of the spectrum envelope and the excitation signal. The spectrum envelope determines the phonetics of the pronounced sound and characterizes the state of the vocal tract, while the excitation signal characterizes the state of the vocal cords and the pitch (intonation) of vocalized sounds. The advantage of LP is also low computational complexity.

However, despite this, recently, preference has been given to models using a sinusoidal representation of the signal, and this primarily concerns applications that involve the synthesis of a speech signal with modified parameters, such as intonation change, voice conversion, text-to-speech synthesis, and others. This fact can be explained by the point that the LP does not provide efficient methods for parametric processing of the excitation signal and continuous synthesis of the output signal. Each speech fragment (frame) of the signal is a separate independent unit, and during synthesis, there is a problem of matching adjacent frames. An inconsistent change in the envelope of the amplitude and phase spectrum during the transition from frame to frame causes the appearance of audible artifacts. In addition, the estimation of the spectrum envelope using classical LP methods is an averaging over the entire frame, as a result of which its accuracy is limited. The order of the predictor determines the complexity of the model: for low-order predictors, the spectrum envelope estimate is overly smoothed, while for high-order predictors, the accuracy becomes selective. For points of the spectrum corresponding to the harmonics of the fundamental tone, the accuracy increases, and for all other points it decreases. The optimal order of the predictor depends on the pitch of the voice, but even in the most favorable case, the accuracy of the spectrum envelope estimate has an error leading to audible distortion.

The usage of cepstral coefficients for modeling speech signals is also a classical approach. The most well developed speech modeling system using cepstral coefficients is TANDEM-STRAIGHT [33], [34]. Just as for the classical methods of analysis based on LP, when estimating the cepstral coefficients, it is assumed that the signal is stationary over the observation interval. Estimation of the envelope of the amplitude spectrum requires smoothing and is also not accurate enough compared to models based on sinusoidal parameters (Figure 6).

Due to its wide capabilities, the hybrid model based on sinusoidal parameters is the most preferred for usage in most practical cases. Nevertheless, to overcome its existing limitations associated with the complexity of estimating parameters, their interpretation in the form of specific speech features (vocal tract parameters, excitation sequence), the development of special modeling methods is required.

Depending on the application, processing of a speech signal using a particular model usually includes analysis (determining the model parameters), modification (changing the model parameters depending on the purpose of the application), and synthesis (forming a new signal from the changed model parameters). Thus, to ensure the highest practical significance, the developed modeling methods should include tools for analysis, processing of

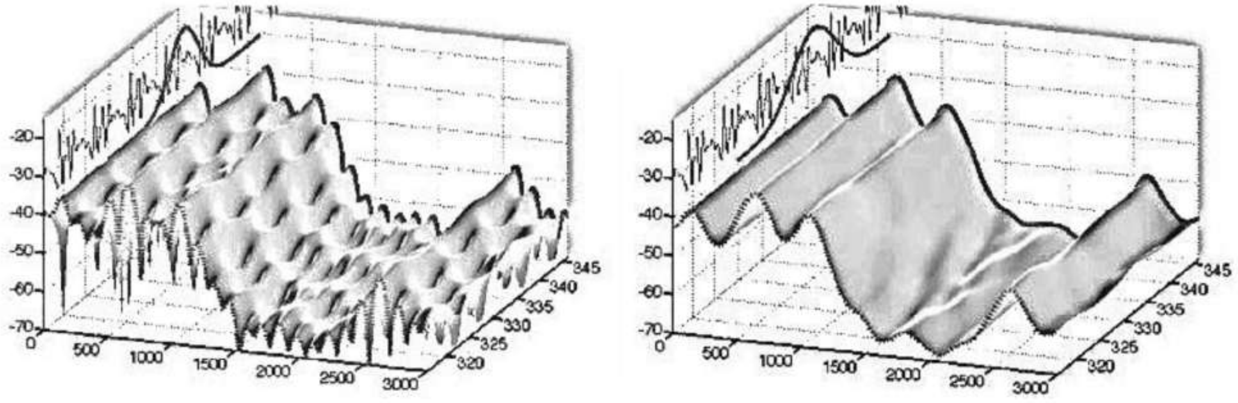


Figure 6. A conventional spectrogram (left) vs TANDEM-STRAIGHT spectrogram (right)

parameters, and synthesis.

Solving many modern applied problems requires not only the ability to describe a speech signal or the phonation process but also the usage of high-level speech features that determine the speaker's personal voice, expression, phonetics, etc. Such problems include voice conversion, text-to-speech synthesis, speaker verification, and many others. High-level speech modeling is a very complex subject domain, since it requires the usage of intelligent models and machine learning methods. At the moment, there is no single universal method used for different applications.

The vast majority of high-level speech models used in practice are problem-oriented and can only be used to solve one, highly specialized problem. The main mathematical tools used are statistical and probabilistic models.

parametric signal model

⇒ *definition**:
[mathematical expression used to represent signal samples in the time or frequency domain]

⊃ *parametric model of the speech signal*

⇒ *definition**:
[mathematical description of signal features that are specific to speech and associated with the phonation process (such as frequency of the main tone, excitation sequence, and amplitude spectrum envelope)]

⇒ *note**:
[The main speech signal models include: models based on linear prediction; based on the cepstral representation; sinusoidal and hybrid models. Among the hybrid models, the harmonic + noise model is the most well known.]

VI. Conclusion

In the article, the ideas underlying the original approach to designing audio interfaces of ICS based on ontological design and formalization of a concepts system from the relevant subject domain, using the OSTIS Technology, is represented. The main principles underlying this approach, as well as their distinctive features from the generally accepted ones, are outlined.

The following main factors can be attributed to the limitations of the proposed approach: it is obvious that in order to achieve the purpose and implement the problems of formalizing any subject domain, including audio interfaces, first of all, a large number of sources of knowledge are required to replenish them. To overcome this problem, it is necessary to involve a large number of experts with appropriate competencies and knowledge in the subject or to develop mechanisms for reliable automatic extraction of this knowledge from available sources.

Direct access to the knowledge of experts is very limited, since it requires significant efforts to select a representative sample of such experts, build effective and interoperable relationships between the parties of the process, which often depends on a large number of subjective factors, and, accordingly, requires a large amount of time and material resources.

It is known that a significant amount of information accumulated by mankind is stored in the form of natural language texts. The process of extracting this information and its representing in a formalized form – in the form of knowledge – also looks non-trivial.

Based on the nature of these problems, according to the authors, the following main directions for overcoming them are seen and, as a result, two main strategies for developing the proposed approach are:

1) Creation of specialized tools for experts working in the domain of audio and voice interfaces for formalizing and representing knowledge from a given subject domain, fixing them in the form of standards of a single form. Such tools should have qualitatively new functionality providing a high level of compatibility and interoperability in the process of accumulation and standardization of knowledge, so that the experts themselves would be interested in the application and wide distribution of this technology for knowledge representation. This item is one of the key objectives of the technology and the OSTIS standard.
2) Creation of automated and automatic means of extracting knowledge from existing sources of information, primarily natural language texts. The types of documents that contain already structured and partly formalized information are primarily standards, protocols, request for comments (RFC), instructions, etc. Therefore, the process of automating knowledge extraction should be aimed primarily at formalizing the existing industry standards for the development of audio interfaces, systems for processing and encoding audio information, speech signal processing systems, such as the standards of the International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers (IEEE), and (Audio Engineering Society) AES series [35]–[39].

The implementation of the approach proposed in the work will ensure the properties of unification, semantic compatibility, and interoperability in the development of audio and voice interfaces (a kind of analogue of the OSI/ISO model in the field of designing ICS interfaces), which ultimately will significantly reduce costs when creating next-generation intelligent computer systems for solving complex problems.

REFERENCES

[1] C. Pearl, *Designing voice user interfaces: principles of conversational experiences*. O'Reilly Media, Inc., 2016.

[2] N. Chen, C. You, and Y. Zou, "Self-supervised dialogue learning for spoken conversational question answering," in *Proc. Interspeech 2021*, 2021, pp. 231–235.

[3] L. Lu, H.-J. Zhang, and H. Jiang, "Content analysis for audio classification and segmentation," *IEEE Transactions on speech and audio processing*, vol. 10, no. 7, pp. 504–516, 2002.

[4] E. Fernandes. (2022) Speech and voice recognition market / verified market research. [Online]. Available: https://www.verifiedmarketresearch.com/download-sample/?rid=4077

[5] J. R. Bellegarda, "Spoken language understanding for natural interaction: The siri experience," *Natural interaction with robots, knowbots and smartphones*, pp. 3–14, 2014.

[6] J. Lemley, S. Bazrafkan, and P. Corcoran, "Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision." *IEEE Consumer Electronics Magazine*, vol. 6, no. 2, pp. 48–56, 2017.

[7] M. B. Hoy, "Alexa, Siri, Cortana, and more: an introduction to voice assistants," *Medical reference services quarterly*, vol. 37, no. 1, pp. 81–88, 2018.

[8] S. Schoolar. Scientific papers search results by keyword "speech technology" and data range filter 2017-2022. [Online]. Available: https://www.semanticscholar.org/search?year%5B0%5D=2017&year%5B1%5D=2022&q=speech%20technology&sort=relevance

[9] V. Popov, S. Kamenev, M. Kudinov, S. Repyevsky, T. Sadekova, V. Bushaev, V. Kryzhanovskiy, and D. Parkhomenko, "Fast and lightweight on-device tts with Tacotron2 and LPCNet," in *Proc. Interspeech*, 2020, pp. 220–224.

[10] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

[11] P. Deepa and R. Khilar, "A report on voice recognition system: Techniques, methodologies and challenges using deep neural network," in *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2021, pp. 1–5.

[12] Alpha Cephei Inc. VOSK is a speech recognition toolkit. [Online]. Available: https://alphacephei.com/vosk/

[13] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," Tech. Rep., Technical report, OpenAI, Tech. Rep., 2022.

[14] V. Delić, Z. Perić, M. Sečujski, N. Jakovljević, J. Nikolić, D. Mišković, N. Simić, S. Suzić, and T. Delić, "Speech technology progress based on new machine learning paradigm." *Computational intelligence and neuroscience*, 2019.

[15] V. Golenkov, N. Guliakina, and D. Shunkevich, *Otkrytaja tehnologija ontologicheskogo proektirovanija, proizvodstva i jekspluatacii semanticheski sovmestimyh gibridnyh intellektual'nyh komp'juternyh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems]*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[16] V. Golenkov, N. Guliakina, I. Davydenko, and A. Eremeev, "Methods and tools for ensuring compatibility of computer systems," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2019, pp. 25–52.

[17] I. Davydenko, "Ontologicheskoe proektirovanie baz znanij [ontology-based knowledge base design]," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2017, pp. 57–72.

[18] D. Shunkevich, "Agentno-orientirovannye reshateli zadach intellektual'nyh sistem [Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems]," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2018, pp. 119–132.

[19] V. Zahariev, N. Hubarevich, and E. Azarov, "Semantic analysis of voice messages based on a formalized context," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2019, pp. 103–112.

[20] V. Zahariev, D. Shunkevich, S. Nikiforov, and E. Azarov, "Intelligent Voice Assistant Based on Open Semantic Technology," in *Open Semantic Technologies for Intelligent System*, V. Golenkov, V. Krasnoproshin, and V. Golovko, Eds. Cham: Springer International Publishing, 2020, pp. 121–145.

[21] V. Zahariev, S. Nikiforov, and E. Azarov, "Conversational speech analysis based on the formalized representation of the mental lexicon," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2021, pp. 141–168.

[22] A. Kroshchanka, V. Golovko, E. Mikhno, M. Kovalev, V. Zahariev, and A. Zagoskij, "A Neural-Symbolic Approach to Computer Vision," in *Open Semantic Technologies for Intelligent System*, V. Golenkov, V. Krasnoproshin, V. Golovko, and D.Shunkevich, Eds. Cham: Springer International Publishing, 2022, pp. 282–309.

[23] O. Räsänen. Linguistic structure of speech. [Online]. Available: https://wiki.aalto.fi/display/ITSP/Linguistic+structure+of+speech

[24] B. Lobanov and O. Eliseeva, *Rechevoj interfejs intellektual'nyh sistem: uchebnoe posobie [Speech User Interface for Intelligent Systems: Tutorial]*. Minsk: BSUIR, Minsk, 2006.

[25] J. Laver, *Principles of phonetics*. Cambridge university press, 1994.

[26] X. Serra, *A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition*. Stanford University, 1990.

[27] D. W. Griffin and J. S. Lim, "Multiband excitation vocoder," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 36, no. 8, pp. 1223–1235, 1988.

[28] A. Petrovsky, E. Azarov, and A. Petrovsky, "Hybrid signal decomposition based on instantaneous harmonic parameters and perceptually motivated wavelet packets for scalable audio coding," *Signal processing*, vol. 91, no. 6, pp. 1489–1504, 2011.

[29] E. Azarov, M. Vashkevich, and A. A. Petrovsky, "Instantaneous harmonic representation of speech using multicomponent sinusoidal excitation." in *INTERSPEECH*, 2013, pp. 1697–1701.

[30] J. Laroche, Y. Stylianou, and E. Moulines, "HNS: Speech modification based on a harmonic+ noise model," in *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2. IEEE, 1993, pp. 550–553.

[31] R. McAulay and T. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.

[32] G. Degottex, P. Lanchantin, A. Roebel, and X. Rodet, "Mixed source model and its adapted vocal tract filter estimate for voice transformation and synthesis," *Speech Communication*, vol. 55, no. 2, pp. 278–294, 2013.

[33] H. Kawahara, "Exploration of the other aspect of vocoder revisited: Az straight, tandem-straight and morphing," in *Seventh ISCA Workshop on Speech Synthesis*, 2010.

[34] H. Kawahara, T. Takahashi, M. Morise, and H. Banno, "Development of exploratory research tools based on tandem-straight," in *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*. Asia-Pacific Signal and Information Processing Association, 2009, 2009, pp. 111–120.

[35] "ISO/IEC 14496-3:2005. Information technology — Coding of audio-visual objects — Part 3: Audio," International Organization for Standardization, Geneva, CH, Standard, 2005.

[36] "ISO/IEC 23003-3:2020 Information technology — MPEG audio technologies — Part 3: Unified speech and audio coding," International Organization for Standardization, Geneva, CH, Standard, 2020.

[37] "IEEE 1857.8-2020 - IEEE Standard for Second Generation Audio Coding," Institute of Electrical and Electronics Engineers, Standard, 2020.

[38] "IEC 62087-2:2015. Audio, video, and related equipment - Determination of power consumption - Part 2: Signals and media," International Electrotechnical Commission, Standard, 2015.

[39] "AES Tech 3250-2004 Specification of the digital audio interface (AES/EBU)," Audio Engineering Society, Standard, 2004.

Аудио-интерфейс интеллектуальных компьютерных систем нового поколения

Захарьев В. А., Жаксылык К. Ж., Лихачев Д. С., Петровский Н. А., Вашкевич М. И., Азаров И. С.

Работа посвящена рассмотрению вопросов создания аудио- и речевых интерфейсов для интеллектуальных компьютерных систем нового поколения. Предлагается использование подхода на основе онтологического проектирования и формализации системы понятий из предметной области аудиоинтерфейсов посредством технологии OSTIS. Изложены основные идеи, лежащие в основе данного подхода, а также особенности, отличающие их от общепринятых.

Суть подхода заключается в рассмотрении процесса проектирования аудио интерфейса как интерфейсной подсистемы в рамках общего процесса разработки интеллектуальной компьютерной системы и построении её формальной логико-семантической модели.

Для достижения поставленной цели предлагается прибегнуть к подходу на основе принципов лежащих в основе "Стандарта открытой технологии онтологического проектирования, производства и эксплуатации семантически совместимых гибридных интеллектуальных компьютерных систем"или кратко "Стандарта технологии OSTIS" [15].

Для создания подобной модели интеллектуальной компьютерных систем нового поколения необходимо:

- произвести декомпозицию информационной компьютерной системы на компоненты. Качество декомпозии при этом определяется простотой последующего синтеза общей формальной модели из формальных моделей выделенных компонентов.
- провести конвергенцию выделенных компонентов в целях построения совместимых формальных моделей этих компонентов;
- провести интеграцию построенных формальных моделей выделенных компонентов и получить общую формальную модель.

Показано, что в перспективе использование данного подхода может обеспечить свойства унификации, семантической совместимости и интероперабельности при разработке аудио- и речевых интерфейсов, что в итоге позволит существенным образом сократить издержки при создании интеллектуальных компьютерных систем нового поколения для решения комплексных задач.

Received 30.10.2022

3D representation of objects in new generation intelligent computer systems

Katsiaryna Halavataya
Belarusian State University
Minsk, Belarus
Email: kat.golovataya@gmail.com

Aliaksandr Halavaty
Belarusian State University
Minsk, Belarus
Email: alex.halavatyi@gmail.com

***Abstract*—This article is dedicated to the issues of constructing and using a three-dimensional representation in various tasks of applied intelligent systems, as well as corresponding systems of spatial positioning and orientation. The description of the representation itself, as well as the principles of its construction, is implemented within the knowledge base of the OSTIS system, which allows for deep integration of various tasks and methods, and also subsequently leads to an increased degree of convergence of various research domains.**

***Keywords*—3D representation, 3D reconstruction, knowledge base**

I. Introduction

For the interaction of an intelligent computer system with objects of the external environment in applied tasks, it is necessary to create an internal representation of these objects. One of the types of such an internal representation can be a description of objects in the form of a three-dimensional model. At the same time, the formation of such an internal representation belongs to the class of tasks for analyzing sensory information and requires determining the exact location of the object, on the basis of which application systems can implement various interaction scenarios. In accordance with this, systems of spatial orientation and three-dimensional reconstruction, capable of forming and processing such representations, are of particular importance.

Application areas that require solving these problems include [1]:

- Intelligent robotic systems. Examples of tasks: environment analysis, motion trajectories building, object three-dimensional coordinates estimation.
- Intelligent production control systems. Examples of tasks: object deformation and structural changes analysis, non-contact dimension measurement for objects of arbitrary scale and configuration, production process control.
- Intelligent systems of complex medical monitoring and service. Examples of tasks: examination result analysis, tracking disease development dynamics, treatment planning.
- Scientific research.
- Other applied areas (architecture, cartography, etc.).

The subject area of the three-dimensional object representation concerns both the description of the object itself and the methods for obtaining this description. Based on these representations, the following classes of problems can be solved:

- building a three-dimensional representation of an object, group of objects or environment,
- determining the size of an object, including calculation of deviations from a given template or parameters, for example, in medical diagnostic systems,
- carrying out additional constructions that further refine the created generated three-dimensional representation,
- building a movement trajectory,
- etc.

Despite the fact that some of the above tasks require additional steps, they are all based on obtaining a three-dimensional object representation.

Thus, the declarative formulation of the problem of three-dimensional reconstruction is to obtain an internal representation of an object belonging to the class of three-dimensional representations.

II. Analysis of existing approaches to 3D reconstruction

At the moment, there exists a large number of methods that operate with different concepts: photogrammetric restoration from photo / video recording, radio frequency methods in different ranges, magnetic and inertial methods, neural network analysis, etc. For example, artificial neural networks that solve the problem of three-dimensional reconstruction can take individual images, image sets, panoramic and stereoscopic images, a combination of images and data from various types of sensors, sets of key points, a voxel cloud as input. For each method, a set of characteristics of the external environment (room, lighting, presence of movement) and input representation (size, type of surface) can be defined, within which this method is correct and demonstrates the best results for one of the target criteria. The resulting internal representation may also differ: some of the methods allow to restore the internal structure, others - only the surface (external shape) of the object.

In addition, in the tasks of analyzing sensory information, it is generally possible to install several different types of sensors, but they must, firstly, be suitable for studying this type of object, and, secondly, the information obtained must complement each other (increase the level of detail, resolution, accuracy, etc.) - that is, the system must be able to adapt to a specific task and external conditions.

All these factors impose serious restrictions on the possibility of using various methods of three-dimensional reconstruction in solving a specific applied problem, which must be taken into account when designing a system.

The problems of the existing solutions include:

- Lack of consistency of concept systems and descriptions of methods in various sources. There are different descriptions and terminologies for the same methods and their modifications, and difficulties and misunderstandings constantly arise because of this.
- Lack of binding and insufficient attention to the issues of convergence of the subject area of three-dimensional reconstruction with the subject area of the formation of three-dimensional scenes and environments of user interaction with the three-dimensional environment, for example, in systems of three-dimensional modeling, virtual and augmented reality.
- High complexity of developing applied systems using 3D reconstruction methods, and the need to involve experienced and highly qualified developers in solving relevant problems.
- Lack of integrated design technology. Despite the abundance of algorithms and methods, the analysis of their applicability to various types of applied problems is extremely superficial. As a result, in most cases the best method is chosen by enumeration or empirically.
- Lack of means for integrating individual components, stages of various methods, various types of data in the description of the object and the resulting internal representations. In addition to the variability of individual actions, usually each method works with its own class of internal representation (a surface specified polygonally, a voxel cloud with a regular grid, a set of individual coordinates in three-dimensional space, etc.).

Thus, the systematization of knowledge in this subject area, as well as the creation of technology for development and automation of the design process of intelligent systems in this area are relevant and currently unresolved tasks.

III. Suggested Approach

The description of the representation itself, as well as the principles of its construction in this paper, are implemented in the form of a knowledge base of the OSTIS system [2]. As part of the formation of a knowledge base and a platform for the development of intelligent systems in this subject area, the following stages have been identified:

- highlighting the semantic representation of three-dimensional scenes;
- systematization of the subject area, existing approaches and establishing links with related areas;
- development of a set of agents that determine the appropriate methods and tools for specific application tasks;
- development of a set of agents that carry out aggregation of different methods in order to clarify or check the parameters of the three-dimensional representation (position) of an object.

The sequence of the main stages of the 3D reconstruction process and their connection with the OSTIS knowledge base is shown in Figure 1. Within the knowledge base, it is proposed to highlight the following blocks:

- description of the characteristics of the observed objects;
- description of types and principles of setting three-dimensional representations;
- a description of the physical principles of operation and specifications of the equipment with which information about the object under study can be collected;
- a set of different methods of reconstruction and localization with limitations and solvers of specific problems;
- methods for evaluating the results of the obtained 3D representations;
- description of the semantic representation of three-dimensional scenes and objects.

Further, we will consider in more detail the main indicated subject areas and onthologies.

IV. Semantic representation of objects and scene

An important component of intelligent systems that use an internal three-dimensional representation is the description of the semantics of this representation. Information about individual points, surfaces, polygons or other primitives does not allow to form a complex idea of the semantic content of this representation, just as individual letters do not allow to evaluate the semantic component of text messages.

The semantic description of an object implies the association of a set of points corresponding to some object in three-dimensional space with some object of the existing knowledge base. Relations in such associations

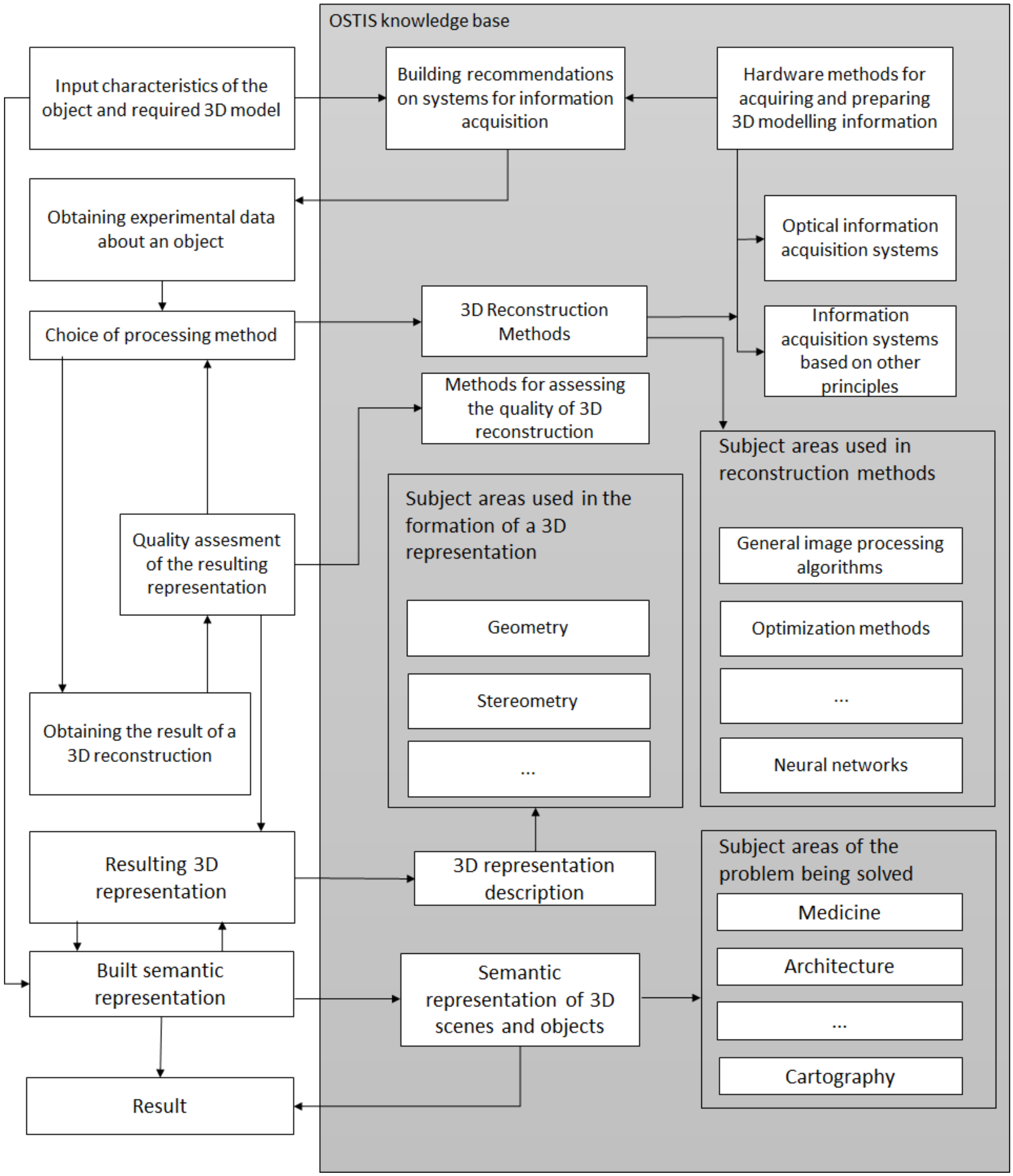

OSTIS knowledge base
Input characteristics of the object and required 3D model
Building recommendations on systems for information acquisition
Hardware methods for acquiring and preparing 3D modelling information
Obtaining experimental data about an object
Optical information acquisition systems
Choice of processing method
3D Reconstruction Methods
Information acquisition systems based on other principles
Methods for assessing the quality of 3D reconstruction
Subject areas used in reconstruction methods
Subject areas used in the formation of a 3D representation
Quality assesment of the resulting representation
General image processing algorithms
Geometry
Optimization methods
Stereometry
...
Obtaining the result of a 3D reconstruction
...
Neural networks
Resulting 3D representation
3D representation description
Subject areas of the problem being solved
Medicine
Built semantic representation
Semantic representation of 3D scenes and objects
Architecture
...
Result
Cartography

Figure 1. Description of the main stages of the 3D reconstruction process and their relationship with the knowledge base

can be represented as “object A is a three-dimensional image of some entity B”, where object A is defined by an internal three-dimensional representation, and entity B is defined by some description in the knowledge base.

Some properties of entity B associated with object A can be naturally inferred from the 3D representation of that object — in particular, information about shape, surface properties, coloration and texturing may already be present in the 3D description. Thus, with the help of additional processing, the three-dimensional representation of object A can be a source of factual information about the related entity B.

Similarly to the semantic description of a separate object, a semantic description of composite objects can be introduced. It should be kept in mind that the semantic content of individual components does not allow to fully determine the semantics of the entire composite object as a whole, therefore, a similar relation must also be set for the entire population. For example, the object "bicycle" can be decomposed into compound objects "chain", "frame", "wheel", etc., however, the set of semantic descriptions of three-dimensional representations of the components separately does not allow one to form or take into account the semantics of the entire compound object as a whole.

The semantic description of a part of an object should contain both information about the base object and additional context regarding the semantic content of the part under consideration. For example, in the videoendoscopic analysis of a section of the esophagus, information about which part of the esophagus in the body this section belongs to also becomes important.

The scene is a complex composition of several simple or composite objects in some common space, supplemented by data of their relative position. As in the case of composite objects, the semantic description of scenes should include not only descriptions of individual components, but also the semantic content of the emergent properties of all these components in the aggregate.

Thus, the following types of main entities have been identified within the framework of the semantic representation of three-dimensional scenes and objects:

object

:= [a set of points in space connected to each other and having one semantic representation]

compound object

:= [object that allows decomposition into separate individual objects]

object part

:= [a set of points in space belonging to some object, which can be distinguished by geometric or semantic representation]

scene

:= [set of several objects and data about their relative position in space]

For the scene, the semantic characteristics of the visual perception of the scene from the position of some observer placed in it or a machine vision system are important. In this context, the scene can be represented as a two-dimensional projection (or a pair of two-dimensional projections in the case of stereoscopic vision), while the semantics of the descriptions of the original three-dimensional objects and the corresponding parts of the resulting projections may also differ - for example, some objects may be out of view, or be perceived differently by the observer due to the presence of some optical, perspective, or psycho-physiological effects (eg, difference in lighting, optical distortion, various illusions of color or depth perception, eg, Ames room, etc.).

It is important for the semantic description of the scene to include both the definition of belonging of the individual objects of the scene to some entities of the existing knowledge base (e.g. as suggested in [3] and [4]), and a description of the possible relationships between these objects, arising due to their presence in the scene, or due to their pairwise mutual arrangement from the position of some observer. This information can naturally be used as reference properties of objects in the scene. For example, if there are two identical objects of type A and one object of type B in the scene, some logical statement or natural language query can utilize the fact of their relative position, for example, to identify one of them - "the object of type A, which is located to the left of the object of type B ".

- Information about the presence on the scene or the projection of the scene
  - The object is missing from the scene
  - The object is present but not visible in the projection
  - The object is present and partially visible on the projection
  - The object is present and is fully visible on the projection
- Information about the relative position on the stage
  - By height
    - Above another object
    - On the same level as another object
    - Below another object
- Information about the relative position on the scene projection
  - By depth
    - Behind another object
    - In front of another object
  - By height
    - Above another object

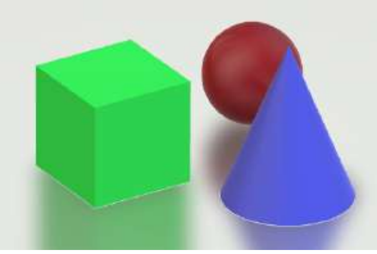


Figure 2. An example of a simple 3D scene rendered as a 2D projection from a specific camera position

        - On the same level as another object
        - Below another object
    - Location along the horizon line
        - To the left of another object
        - To the right of another object
- Information about the relative size of objects on the projection
    - Bigger than another object
    - Same size as another object
    - Less than another object
- Information about the visual similarity of objects
    - Similar or not another object in shape
    - Similar or not another object in colour
    - Similar or not another object in size

An example of a simple three-dimensional scene, as well as its semantic description in terms of the mutual arrangement of objects, is shown in Figures 2 and 3.

V. 3D REPRESENTATION

Systems for positioning, recognition and visualization of objects in the real world are based not only on the qualitative component of the description, but also on the relative spatial position of objects or their individual parts. In accordance with the perception of the surrounding world by a person, a three-dimensional representation is the most informative, although not mandatory. The resulting two-dimensional images used in many tasks are projections of three-dimensional space. Therefore, in this paper, a three-dimensional representation of objects, including a class of descriptions containing information about the relative position of objects or their parts in three coordinates, is chosen as the maximum class of the study objects.

```
3D representation
⇒   split*:
    {•   surface representation
         ⇒   includes*:
             {•   polygon meshes
              •   NURBS surfaces
              •   separation surfaces
              •   surfaces based on T-splines
             }
     •   voxel representation
         ⇒   includes*:
             {•   point cloud
                  ⊃   depth map
              •   structural mesh
             }
     •   special representations
         ⇒   includes*:
             {•   video 360
             }
    }
```

VI. 3D RECONSTRUCTION

A three-dimensional reconstruction allows obtaining a three-dimensional representation based on other data. Three-dimensional reconstruction is the task of determining the true form of objects in three-dimensional space based on information about these objects obtained as a result of measurements, observations or experiments.

Each of the methods of three-dimensional reconstruction can be characterized, in addition to the physical principle of operation, also by the resolution, the type of input data, the size and internal structure of the reconstructed objects, etc. The full description is a non-hierarchical ontological model [5]. For further interaction of agents with a given structure, all these descriptions are mapped onto some characteristics. Characteristics can apply both to a separate method and to a group of methods, for example, the resolution can be common to the entire subclass of electromagnetic wave methods. These characteristics should be dynamically obtained by agents from the knowledge representation itself, which allows supplementing and modifying the overall structure. For convenience of presentation, these characteristics can be identified in the specification of the method, on the basis of which the scope and possibility of its application can be described. It makes it possible to use methods for solving specific applied problems. Within the framework of the specification (and, accordingly, the structure of the representation of methods in the knowledge base), the following can be specified:

- type of possible input parameters,
- the output representation type, in this case the corresponding 3D representation;
- working hours;
- resolution of the method;
- distance from the object to the camera;
- type of reconstructed object;
- scene composition (separate object, group of objects, surrounding space);
- surface type (gloss, transparency, color);
- the presence of an internal structure;
- object size.

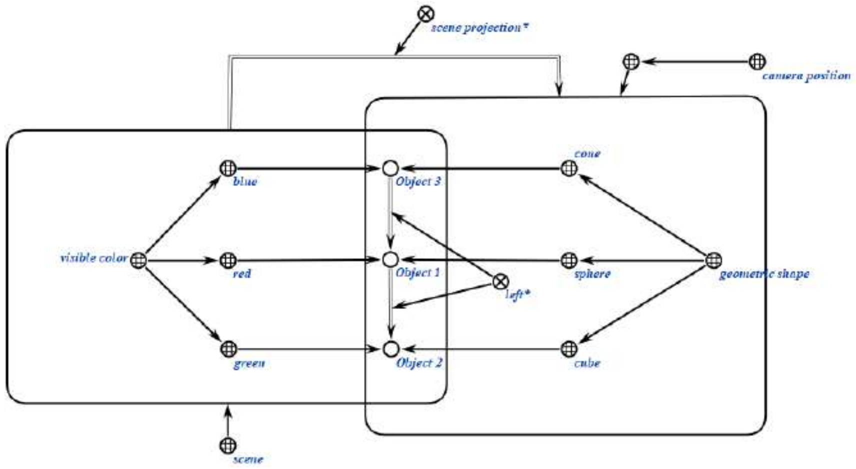


Figure 3. Semantic description of some relations of mutual arrangement of objects in the scene from the position of the observer

The described specification can be interpreted as an intermediate relation for each method. With such a description for the intermediate steps, this approach also allows to combine methods. For example, radio frequency methods do not make it possible to build a depth map, but allow to get the position of the camera in the global coordinate system at a specific point in time. [6].

3D reconstruction methods
⇒ *includes**:
{• *methods of photogrammetric reconstruction*
  ⇒ *divided by the relative position of the object and the camera into**:
    {• *mobile systems*
    • *macrophotogrammetry*
    • *satellite photogrammetry*
    • *aerophotogrammetry*
    • *terrestrial photogrammetry*
    • *near photogrammetry*
    }
  ⇒ *divided according to the type of input information into**:
    {• *single image*
    • *stereo images*
    • *multiframe*
    }
• *methods of tomographic reconstruction*
  ⇒ *includes**:
    {• *reconstruction based on Fourier projections*
    • *back projection algorithm*
    • *iterative reconstruction algorithm*
      ⇒ *includes**:
        {• *ART*
        • *SART*
        • *SAMV*
        }
    • *conical beam reconstruction*
    • *reconstruction based on deep learning methods*
    }
• *structured highlighting methods*
  ⇒ *includes**:
    {• *methods based on light sections*
    • *methods based on band projections*
    • *methods based on phase shift*
    }
• *methods for estimating the reflected signal*
  ⇒ *includes**:
    {• *distance measurement by optical modulation methods*
    • *pulse modulation*
    }
• *focus evaluation methods*
  ⇒ *includes**:
    {• *evaluation methods from focus*
    • *defocus evaluation methods*
    }
• *methods based on the Fourier projection theorem*
• *neural network models*
}

Many 3D reconstruction systems are based on solving the problem of local positioning. It should be noted that, in general, the problem of local positioning is wider in scope; however, it is usually considered relative to the observer, and not to the object.

Characteristics of positioning systems:

- positioning accuracy,
- positioning authenticity,
- polling frequency,
- reliability,
- size.

An ontological description of positioning methods can be obtained on the basis of a subdivision, both according to the physical principle of obtaining initial data, and according to the calculation method.

VII. Ontology of domain actions

At a higher level, any 3D reconstruction method can be represented as procedural knowledge, such as a series of steps that transform the 3D reconstruction input into a final 3D representation.

A common feature of many methods of three-dimensional reconstruction is the use of an intermediate (or final) representation of the objects of the surrounding world in three coordinates. In other words, the 3D reconstruction method can be represented as a sequence of actions to form a set of elements characterized by coordinates in a common 3D space, which, if necessary, can be further built up to surfaces, combined with 2D representations, etc. to form the desired output representation:

$$r : Im(R^3) \to O \tag{1}$$

where I is the input data, R^3 is the common three-dimensional Cartesian space, $m(R^3)$ is the description of the element in the coordinate system of the common three-dimensional space, O is the output three-dimensional representation. Most often, individual points of three-dimensional space act as elements of an intermediate representation - in this case, such a representation is called a point cloud.

Curve segments, analytically defined surfaces, polygons and other types of objects can also act as elements.

The sources of data on three-dimensional coordinates for the intermediate representation can be:

- Direct absolute values of three-dimensional coordinates of points, i.e. in this case the input I is a set of points R^3. This is typical for all methods that allow building a scene depth map, because representation in the form of a depth map with known coordinates of the position of the observer and the focal length of the map allows to determine three-dimensional coordinates of any point on it.
- Data from which, with the help of some additional processing, the values of the three-dimensional coordinates of individual points can be obtained. Such representations tend to be much more common and easier to obtain.

It should be noted that different data sources can be used together when forming an intermediate representation, if each of the sources has some binding to a common coordinate system.

A. Actions for generating an intermediate 3D representation

Many remote sensing methods rely on the availability of a priori information about the three-dimensional coordinates of the object under study, from which a point cloud can be built for an intermediate three-dimensional representation. These include methods that allow you to estimate the distance from the measuring device to the object. However, the use of these methods requires more sophisticated equipment, which can be difficult to apply in some scenarios.

In this regard, it's possible to isolate a separate category of research methods - group of methods for forming an intermediate representation as a point cloud, based on more traditional types of information. These include methods for generating from a single image, a stereopair, a set of images, or a video sequence, which can also use information about the optical system of the camera that was used to obtain the image.

Thus, the following types of input data can be considered:

- A static image or set of static images;
- Video sequence (a set of static images with a timestamp);
- Stereopair (2 static images with optical system parameters) or a set of stereopairs;
- Stereo video sequence (a set of stereopairs with a timestamp).
- Information about the optical system of the camera.

In this case, a sparse point cloud of three-dimensional space acts as an output representation, with each of the points of this space bound to one or more points of the original input images.

The formation of a sparse point cloud includes the following steps, for each of which the specified parameters that can be determined:

- Preprocessing
- Keypoint detection
  - detector algorithm
- Keypoint matching
  - descriptor algorithm or optical flow algorithm
  - thinning
- Evaluation of camera positions
  - projection model
  - bundle evaluation algorithm
- Postprocessing

Detection and matching of key points allows to determine the points belonging to the same object of the studied three-dimensional space, if there are a sufficient number of images. At the stage of estimating the position of the camera, a mathematical projection model is used that specifies the relationship between the two-dimensional coordinates of a point in the image and the corresponding three-dimensional coordinates in the modeling space; at the same stage, empirical depth estimation can be carried out, for example, using neural network methods. Each keypoint match, described mathematically using the projection model, is further used in the pose estimation algorithm in order to restore the camera positions in three-dimensional space for each of the analyzed images, and also to determine the distances from the cameras to the points, based on some criterion for minimizing the back-projection error.

For example, the classical Structure from Motion three-dimensional reconstruction method from several input images within the presented pipeline can be described by the following characteristics:

- Preprocessing – conversion to grayscale;
- Detector algorithm - SIFT, SURF, FAST;
- Descriptor algorithm - SIFT, SURF, ORB;
- Thinning - RANSAC;
- Projection model – central projection;
- Pose estimation algorithm - global bundle adjustment method with Levenberg-Marquardt optimization.

As another example, consider the ORB-SLAM method of simultaneous localization and mapping [7], using a video sequence as an input representation:

- Preprocessing - frame thinning;
- Detector algorithm - FAST;
- Descriptor algorithm – ORB + Lucas-Kanade optical flow method;
- Thinning – RANSAC, motion invariant thinning;
- Projection model – central projection;
- Pose estimation algorithm - incremental bundle adjustment with a fixed camera position and a fixed position of key points between frames, mapping method;
- Post-processing - trajectory refinement and loop detection.

An ontological description of the presented sequence of actions, as well as an example of a specific algorithm implemented according to this sequence of actions, are presented in the figures.

Since each of the proposed stages is described as a functional mapping, it is assumed that stages are added, removed or modified during processing if the correspondence between the types of input and output representations is observed in the context of the problem being solved.

B. Actions for generating the final 3D representation

As already mentioned, in some problems a sparse point cloud in three-dimensional space can be a sufficient representation, and can be considered the result of a three-dimensional reconstruction algorithm. Also quite popular is the representation in the form of a dense colored point cloud.

However, in many problems such a representation is not enough, so it is possible to distinguish a class of actions when generating a more complex three-dimensional representation, depending on the type of output representation required. The input representation for this stage is a sparse point cloud, as well as additional information about the relationship of specific points in the cloud with the original representations.

The description of the methods for generating the final three-dimensional representation can be represented as a combination of the following stages, with the corresponding parameters:

- Point cloud density increase
  - algorithm for density increase
  - link to source data
- Surface Shaping
  - surface type
  - surface generation algorithm
- Refine surfaces
  - surface refinement algorithm
  - link to source data
- Surface texturing
  - texture method
  - link to source data
  - conflict resolution

At the point cloud density increase stage, information about the relationship between the 3D coordinates of the sparse cloud points and the source data is used to transfer additional points directly from the source representation to the 3D model. Next, by analyzing the obtained dense point cloud and the initial data, a rough estimate of the final three-dimensional surface is formed in the form of some three-dimensional primitive, usually by forming a polygonal mesh by combining the nearest points into triangles. At the stage of refinement of surfaces, smoothing, thinning and merging of primitives obtained at the previous stage can occur, based on some information from the source data. Finally, at the texturing stage, the original representation is transferred to the constructed 3D model to ensure its realism; also at this stage, conflicts are resolved to select the correct texturing strategy in the presence of several conflicting sources of information about the texture.

For example, the surface reconstruction algorithm proposed in the framework of the currently most popular implementation of bundle adjustment method can be described as the following set of parameters:

- cloud density increase algorithm - transfer of neighborhing keypoints
- surface type - polygonal with rectangular polygons

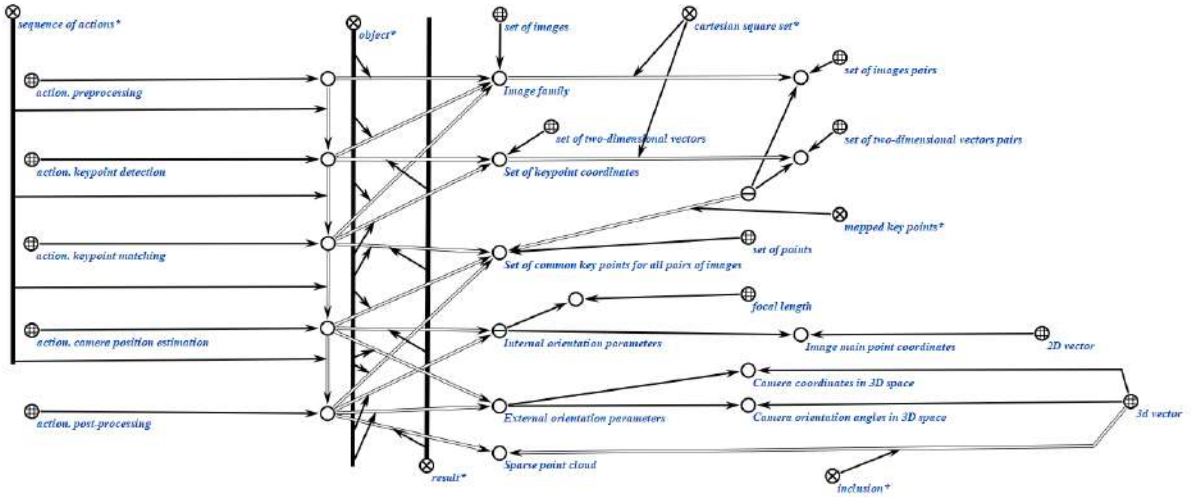

sequence of actions*
object*
set of images
cartesian square set*
set of images pairs
action. preprocessing
Image family
set of two-dimensional vectors
set of two-dimensional vectors pairs
action. keypoint detection
Set of keypoint coordinates
mapped key points*
set of points
action. keypoint matching
Set of common key points for all pairs of images
focal length
action. camera position estimation
Internal orientation parameters
Image main point coordinates
2D vector
Camera coordinates in 3D space
action. post-processing
External orientation parameters
Camera orientation angles in 3D space
3d vector
result*
Sparse point cloud
inclusion*

Figure 4. Ontological description of the sequence of actions for constructing an intermediate three-dimensional representation in the form of a sparse point cloud

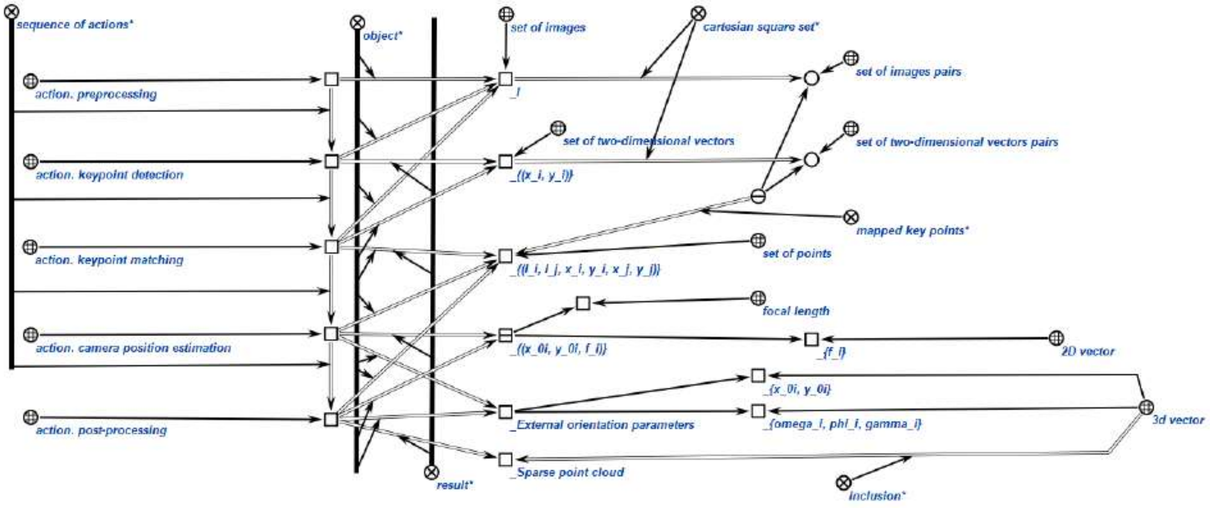


Figure 5. An example of an algorithm for constructing a sparse point cloud based on the presented description

- surface generation algorithm – estimation of camera translations using Delaunay triangulation, estimation of planar camera translations, interpolation between camera positions, manual adjustment
- surface refinement algorithm - missing
- texturing method - direct transfer from source images
- method of resolving texture conflicts - alpha-blending proportional to the distance to the point.

C. Actions for the selecting stages and generation of the algorithm

A detailed description of the structure of the methods is not sufficient for the direct implementation of a three-dimensional reconstruction according to some algorithm. Since there exists a large number of implementations for each of the stages of the final pipeline, the problem also arises of the optimal choice from the set of implementations of each of the stages for the most effective solution of the task.

An agent-based approach to information processing can be used both to select the stages of the 3D reconstruction algorithm and to directly implement it [8]. Agents communicate by accessing a common representation in the knowledge base, which generates a number of questions that specify further actions.

As the main questions on the basis of which the generation of the algorithm can be carried out, the following can be distinguished:

- What input will be used for the reconstruction?
  - Is it possible to determine the distance to a point in three-dimensional space from the input data, or directly its three-dimensional coordinates?

- Will images be used as data?
    - ∗ Are the optical parameters of the camera known?
    - ∗ Are the positions or orientations of the camera in space known for each image?
    - ∗ Is the set of images a continuous video sequence at a known frame rate?
- If there are several sources of input data, is there information about binding the data they describe to a common coordinate system?

- What type of 3D model should be generated from this data?
  - Is reconstruction of 3D surfaces required?
    - ∗ Can the source data be used to form surface textures?

Based on the proposed approach, both the selection of a method that meets the requirements of the problem and the selection of individual stages of the algorithm can be carried out, for example, by choosing the most optimal keypoint detector and descriptor algorithm [9]. In addition, it is possible to combine 3D representations obtained by different methods [10].

VIII. Conclusion

The article considers an ontological description of the subject area of three-dimensional reconstruction of objects and actions for their processing in knowledge bases based on OSTIS technology. The description is presented in the form of the domain area of the scenes themselves, the methods for obtaining them, and the corresponding actions for processing them.

The advantages of using OSTIS technology for the considered tasks are:

- Introduction of a common system of concepts and descriptions of methods in a unified and consistent form.
- Possibility of convergence of the subject area of 3D reconstruction with the subject area of 3D scene and environment building and adjacent areas.
- Simplification of the development of applied systems using 3D reconstruction methods.
- Ability to build a complex design technology using intelligent agents that consume the proposed description.
- Ability to create integration tools for individual components, stages of various methods and resulting internal representations.

Using the proposed approach, it becomes possible to create intelligent systems that can receive and operate a three-dimensional representation for further processing in applied problems.

Acknowledgment

The authors would like to thank the scientific team of the Department of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for their help in the work and valuable comments.

References

[1] T. Luhmann, S. Robson, S. Kyle, and J. Böhm, *Close-Range Photogrammetry and 3D Imaging: 2nd Edition*. Berlin: De Gruyter, 2018.

[2] V. Golenkov, "Ontology-based design of intelligent systems," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2017, pp. 37–56.

[3] T. Homburg, A. Cramer, L. Raddatz, and H. Mara, "Metadata schema and ontology for capturing and processing of 3d cultural heritage objects," *Heritage Science*, vol. 9, no. 91, pp. 2050–7445, Jul. 2021. [Online]. Available: https://doi.org/10.1186/s40494-021-00561-w

[4] C. Cruz, F. Marzani, and F. Boochs, "Ontology-driven 3d reconstruction of architectural objects." SciTePress, 2007, pp. 47–54. [Online]. Available: https://doi.org/10.5220/0002047300470054

[5] D. Shunkevich, "Ontological approach to the development of hybrid problem solvers for intelligent computer systems," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2021, pp. 63–74.

[6] J. Albertz and M. Wiggenhagen, *Guide for Photogrammetry and Remote Sensing*. Hiedelberg: Wichmann, 2009.

[7] R. Mur-Artal, J. Montiel, and J. Tardos, "Orb-slam: a versatile and accurate monocular slam system." Institute of Electrical and Electronics Engineers, 2015, pp. 1147 – 1163. [Online]. Available: https://doi.org/10.1109/TRO.2015.2463671

[8] D. Shunkevich, "Agentno-orientirovannye reshateli zadach intellektual'nyh sistem [Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems]," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2018, pp. 119–132.

[9] K. Halavataya, "Local feature descriptor indexing for image matching and object detection in real-time applications," in *Pattern Recognition and Information Processing*, M. Lukashevich, Ed. BSUIR, Minsk, 2018, pp. 302–305.

[10] K. Halavataya, K. Kozadaev, and V. Sadau, "Adjusting videoendoscopic 3d reconstruction results using tomographic data," *Computer Optics*, vol. 46, no. 2, pp. 246–251, Mar. 2022. [Online]. Available: https://doi.org/10.18287/2412-6179-CO-910

3D-представление объектов в интеллектуальных компьютерных системах нового поколения

Головатая Е.А., Головатый А.И.

Данная статья посвящена рассмотрению вопросов построения и использования трехмерного представления в различных задачах прикладных интеллектуальных систем, а также соответствующих систем позиционирования и ориентации в пространстве. Описание самого представления, а также принципов его построения осуществляется на основе базы знаний OSTIS-системы, что позволяет проводить глубокую интеграцию различных задач и методов, а также в последствии приводит к повышению степени конвергенции различных направлений.

Received 14.11.2022

Comprehensive library of reusable semantically compatible components of next-generation intelligent computer systems

Maksim Orlov
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: orlovmassimo@gmail.com

***Abstract*—The most important stage in the evolution of any technology is the transition to the component design based on a constantly augmented library of reusable components. In the article, an approach to the design of knowledge-driven systems is considered, focused on the usage of compatible reusable components, which significantly reduces the complexity of developing such systems.**

***Keywords*—Component design of intelligent computer systems; reusable semantically compatible components; knowledge-driven systems; semantic networks, ontology design.**

I. Introduction

As the analysis of modern information technologies shows, along with achievements, they have a number of serious disadvantages associated with the complexity of their development and maintenance. In particular, such disadvantages include the following ones [1], [2], [3]:

- there is no general unified solution to the problem of the semantic compatibility of computer systems, which causes a high complexity of creating complex integrated computer systems;
- there is a variety of semantically equivalent implementations of problem-solving models, duplication of knowledge base and user interface components that differ not in the essence of these components but in the form of representation of the processed information;
- the degree of dependence of computer system architectures on the platforms on which they are implemented is high, which causes the complexity of transferring computer systems to new platforms;
- modern information technologies are not oriented to a wide range of developers of applied computer systems;
- there is a lack of a unified approach to the allocation of reusable components and the formation of libraries of such components, which leads to a high complexity of reusage and integration of previously developed components in new computer systems.

Most of the existing systems are created as self-contained software products that cannot be used as components of other systems. It is necessary to use either the whole system or nothing. A small number of systems support a component-oriented architecture capable of integrating with other systems [4], [5]. However, their integration is possible if the same technologies are used and only when designed by one development team [6].

Repeated re-development of existing technical solutions is conditioned either by the fact that known technical solutions are hardly integrated into the system being developed or by the fact that these technical solutions are difficult to find [7]. This problem is relevant both in general in the field of computer systems development and in the field of knowledge-based systems development, since in systems of this kind the degree of consistency of various knowledge types affects the ability of the system to solve non-trivial problems [8].

To solve these problems, it is proposed to implement a comprehensive library of reusable semantically compatible components of next-generation intelligent computer systems.

The development technology should allow components to be reused, integrated with other components built using both this and other technologies. It should also be open to allow using components by different development teams.

Reusage of ready-made components is widely used in many industries related to the design of various kinds of systems, since it allows reducing the complexity of development and its cost (by minimizing the amount of work due to the absence of the need to develop any component), improving the quality of the created content, and reducing professional requirements for computer system developers [9], [10]. Thus, the transition is made from programming components or entire systems to their design based on ready-made components. The usage of ready-made components assumes that the distributed component is verified, tested, evaluated, and documented, and possible limitations are eliminated or specified and known.

The following problems exist in the implementation of the component approach to the design of next-generation intelligent computer systems [11]:

- incompatibility of components developed within different projects due to the lack of unification in the principles of representing different types of knowledge within the same knowledge base and, as a consequence, the lack of unification in the principles of allocation and specification of reusable components;
- the inability to automatically integrate components into the system without manual user intervention;
- testing, verification, and analysis of the components quality are not carried out; advantages, disadvantages, limitations of components are not identified;
- the development of standards that ensure the compatibility of these components is not being carried out;
- many components use the language of the developer for identification (usually English), and it is assumed that all users will use the same language. However, for many applications, this is unacceptable – identifiers that are understandable only to the developer should be hidden from end users, who should be able to choose the language for the identifiers they see;
- the lack of tools to search for components that meet the specified criteria [12].

The purpose of the work is to create conditions for effective, meaningful, and mass design of next-generation intelligent computer systems and their components by creating an environment for the collection and sharing of components of these systems. Such conditions are realized by unlimited expansion of constantly evolving semantically compatible intelligent computer systems and their components. The spheres where the technology of component design of semantically compatible intelligent systems is applied in practice have no limits.

II. Analysis of existing approaches to solving the problem

At the moment, there is no comprehensive library of reusable semantically compatible components of computer systems in general, aside from intelligent ones. There are some attempts to create libraries of typical methods for traditional computer systems, but such libraries do not solve the above problems.

Traditional solutions include package managers of programming languages and operating systems, as well as separate systems and platforms with built-in components and tools for saving created components.

Library components can be implemented in different programming languages and can also be located in different places, which leads to the fact that a tool is needed in the library to find components and install them [13].

Modern package managers such as npm, pip, apt, maven, poetry, and others have the advantage of being able to resolve conflicts when installing dependent components, but they do not take into account the semantics of components but only install components by identifier [14]. Libraries of such components are only a storage of components, which does not take into account the purpose of components, their advantages and disadvantages, scope of application, hierarchy of components, and other information necessary for the intellectualization of component design of computer systems. This storages are realized as some kind of repository, that is not compatible together [15]. There is no single interacting system to store, analyze and supply reusable components. Similarly, a significant disadvantage of the modern approach is the platform dependency of components. Modern component libraries are focused only on a specific programming language, operating system, or platform.

Based on the Modelica language, a large number of freely available component libraries have been developed, one of which is the Modelica_StateGraph2 library, which includes components for modeling discrete events, reactive and hybrid systems using hierarchical state diagrams [16]. The main disadvantage of Modelica-based systems is the lack of component compatibility and sufficient documentation.

Microsoft Power Apps is a set of applications, services, and connectors, as well as a data platform that provides a development environment for efficiently creating user applications for business. The Power Apps platform provides tools for creating a library of reusable graphical interface components, as well as pre-created text recognition models (reading visiting cards or cheques) and an object detection tool that can be connected to the application being developed [17]. The Power Apps component library is a set of user-created components that can be used in any application. The advantage of the library is that components can configure default properties that can be flexibly edited in any applications that use the components. The disadvantage lies in the lack of semantic compatibility of components, the specification of components; the problem of the presence of semantically equivalent components has not been solved; there is no hierarchy of components and means of searching for these components.

WebProtege is a multi-user web interface that allows editing and storing ontologies in the OWL format in a collaborative environment [18]. This project allows not only creating new ontologies but also loading existing ontologies that are stored on the Stanford University server. The advantage of this project is the automatic error checking in the process of creating ontology objects. This project is an example of an attempt to solve the problem of accumulation, systematization, and reusage of existing solutions, however, the disadvantage of this solution is the isolation of the ontologies being developed. Each developed component has its own hierarchy of concepts, an approach to the allocation of classes and entities that depend on the developers of these ontologies, since within this approach, there is no universal model of knowledge representation, as well as formal specification of components represented in the form of ontologies [19]. Consequently, there is a problem of their semantic incompatibility, which, in turn, leads to the impossibility of reusage of the developed ontologies in the knowledge bases design. This fact is confirmed by the presence on the Stanford University server of a variety of different ontologies on the same topics [20].

The IACPaaS platform is designed to support the development, management, and remote usage of applied and instrumental multi-agent cloud services (primarily intelligent) and their components for various subject domains [21]. The platform provides access to:

- application users (specialists in various subject domains) – to applied services;
- developers of applied and instrumental services and their components – to instrumental services;
- intelligent services managers and management services.

The IACPaaS platform supports:

- the basic technology for the development of applied and specialized instrumental (intelligent) services using the basic instrumental services of the platform that support this technology;
- a variety of specialized technologies for the development of applied and specialized instrumental (intelligent) services, using specialized platform tool services that support these technologies.

The IACPaaS platform also does not contain the means for a unified representation of the components of intelligent computer systems and the means for their specification and automatic integration.

III. Proposed approach

Within this article, it is proposed to take an OSTIS Technology [22] as a basis, the principles of which make it possible to implement a library of semantically compatible components of intelligent computer systems and, accordingly, provide the ability to quickly create knowledge-driven systems using ready-made compatible components.

The systems developed on the basis of the OSTIS Technology are called *ostis-systems*. The *OSTIS Technology* is based on a universal method of semantic representation (encoding) of information in the memory of intelligent computer systems, called an *SC-code*. Texts of the *SC-code* (sc-texts) are unified semantic networks with a basic set-theoretic interpretation, which allows solving the problem of compatibility of various knowledge types. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, depending on orientation, can be *sc-arcs* or *sc-edges*). The *Alphabet of the SC-code* consists of five main elements, on the basis of which SC-code constructions of any complexity are built, including more specific types of sc-elements (for example, new concepts). The memory that stores SC-code constructions is called semantic memory, or *sc-memory*.

Within the technology, several universal variants of visualization of *SC-code* constructions are proposed, such as *SCg-code* (graphic variant), *SCn-code* (nonlinear hypertext variant), *SCs-code* (linear string variant).

Within this article, fragments of structured texts in the SCn code [23] will often be used, which are simultaneously fragments of the source texts of the knowledge base, understandable to both human and machine. This allows making the text more structured and formalized, while maintaining its readability. The symbol ":=" in such texts indicates alternative (synonymous) names of the described entity, revealing in more detail certain of its features.

The basis of the knowledge base within the OSTIS Technology is a hierarchical system of subject domains and ontologies. Based on this, in order to solve the problems set within this article, it is proposed to develop the following system of subject domains and ontologies:

Subject domain of reusable ostis-systems components
⇒ *private subject domain**:
Subject domain and ontology of a comprehensive library of reusable semantically compatible ostis-systems components

Subject domain and ontology of a comprehensive library of reusable semantically compatible ostis-systems components
⇒ *private subject domain**:
- *Subject domain and ontology of the library of reusable components of ostis-systems knowledge bases*
- *Subject domain and ontology of the library of reusable components of ostis-systems problem solvers*
- *Subject domain and ontology of the library of reusable components of ostis-systems interfaces*

In the subject domain and ontology of the library of reusable semantically compatible ostis-systems components, the concepts and principles most common to all child subject domains are described, which are valid for any library of reusable components.

The idea of a component library is not new, but the semantic potency of the ***OSTIS Library*** is significantly higher than for analogues due to the fact that the vast majority of library components are knowledge base components represented in a unified knowledge representation language (*SC-code*). Thus, the OSTIS Library provides a high level of semantic compatibility of components, which leads to a high level of semantic compatibility of ostis-systems using the library of reusable ostis-systems components.

Next, we will consider in more detail the fragments of sc-models of the specified subject domain and ontology.

IV. Concept of a library of reusable ostis-systems components

The basis for the implementation of the component approach within the *OSTIS Technology* is the OSTIS Library. The *OSTIS Metasystem* is focused on the development and practical implementation of methods and tools for component design of semantically compatible intelligent systems, which provides an opportunity to quickly create intelligent systems for various purposes. The OSTIS Metasystem includes the ***OSTIS Metasystem Library***.

library of reusable ostis-systems components
⇒ *abbreviation**:
  [library of ostis-systems components]
:= [library of reusable OSTIS components]
∋ *typical example′*:
  OSTIS Library
    := [Library of reusable ostis-systems components as part of the OSTIS Metasystem]
    := [OSTIS Metasystem Library]
⇐ *combination**:
  {• *library of typical subsystems of ostis-systems*
   • *library of templates for typical ostis-systems components*
   • *library of ostis-platforms*
   • *library of reusable knowledge base components*
   • *library of reusable problem solver components*
   • *library of reusable user interface components*
  }

The constantly expanding OSTIS Library significantly reduces the time for the development of new intelligent computer systems. The library of ostis-systems allows getting rid of duplication of semantically equivalent information components as well as from the variety of forms for the technical implementation of the problem-solving models used.

Currently, a large number of *knowledge bases* in a variety of subject domains have been developed. However, in most cases, each knowledge base is developed separately and independently of the others in the absence of a single unified formal basis for the knowledge representation, as well as common principles for the formation of concept systems for the described subject domain. In this regard, the developed bases are, as a rule, incompatible with each other and are not suitable for reusage. A component-based approach to the development of intelligent computer systems, implemented in the form of a ***library of reusable ostis-systems components***, allows solving the described problems. In the field of development of *problem solvers*, there are also a large number of specific implementations, however, problems of compatibility of different solvers when solving a single problem are hardly considered. Hypothetically, the existence of a universal problem solver combining all known problem-solving methods and ways is possible. However, the usage of such a solver for applied purposes is not advisable. Thus, the most acceptable option is to create a library of compatible components, from which a solver that meets the necessary requirements can later be compiled.

Functionality of the library of reusable ostis-systems components:

- Storing reusable ostis-systems components and their specifications. At the same time, some of the components specified within the library may be physically stored in another place due to the peculiarities of their technical implementation (for example, the source texts of the ostis-platform may be physically stored in a separate repository but be specified as a component in the corresponding library). In this case, the specification of the component within the library has to contain the description of (1) the location of the component and (2) the scenario of its automatic installation in a child ostis-system.
- Viewing available components and their specifications, as well as searching for components by fragments of their specification.
- Storing information about which of the library components and which version are used (have been downloaded) in particular consumer ostis-systems. This is necessary at least to take into account the demand for a particular component, to assess its importance and popularity.
- Systematization of reusable ostis-systems components.
- Providing versioning of reusable ostis-systems components.
- Searching for dependencies between reusable components within the library of components.
- Ensuring automatic updating of components borrowed into the child ostis-systems. This function can be turned on and off upon request of the developers of the child ostis-system. Simultaneous updating of the same components in all systems using it should

not lead to inconsistency between these systems in any context. This requirement may be quite complicated but is essential for the operation of the OSTIS Ecosystem.

library of reusable ostis-systems components
⇒ *generalized decomposition**:
{• *knowledge base of the library of reusable ostis-systems components*
• *problem solver of the library of reusable ostis-systems components*
• *interface of the library of reusable ostis-systems components*
}

A knowledge base of the library of reusable ostis-systems components is a hierarchy of reusable ostis-systems components and their specifications, as well as a system of concepts necessary for the specification, installation, and search of components.

A problem solver of the library of reusable ostis-systems components implements the functionality of the ostis-systems library described above.

An interface of the ostis-systems library provides access to reusable components and features of the ostis-systems library for the user and other systems. The interface allows the *manager of reusable ostis-systems components*, which is part of a child ostis-system, to connect to the library of reusable ostis-systems components and use its functionality, that is, for example, to access the specification of components and install selected components in a child ostis-system, get information about the available versions of the component, its dependencies, etc.

V. Place of the OSTIS Library in the architecture of the OSTIS Ecosystem

Developers of any ostis-system can include a library in its structure, which will allow them to accumulate and distribute the results of their activities among other participants of the *OSTIS Ecosystem* in the form of ***reusable components***. The decision to include the component in the library is made by the expert community of developers responsible for the quality of this library. Component verification can be automated. Within the *OSTIS Ecosystem*, there are many libraries of reusable ostis-systems components that are subsystems of the corresponding maternal ostis-systems. The main library of reusable ostis-systems components is the *OSTIS Metasystem Library*. The *OSTIS Metasystem* acts as a ***maternal system*** for all ostis-systems being developed, since it contains all the basic components (Figure *1*). The maternal system is responsible for some class of components and is a SAD for this class, contains methods for the development of such components, their classification, detailed explanations for all subclasses of components. Thus, a hierarchy of maternal ostis-systems is formed. The maternal ostis-system, in turn, can be a child ostis-system for some other ostis-system, borrowing components from the library that is part of this other ostis-system.

Publishing a component to a certified library requires additional effort from the developer to ensure the quality of the component specification and description of its relationship with other components, however, provides the following benefits of using the OSTIS Ecosystem infrastructure.

- Downloads of components registered in a certified library are captured and tracked. the quality and importance of the component is automatically proven by the number of its downloads, this is visible to all members of the community. Thus, the rating of the developer is formed, it becomes more popular and in demand. Registering a component in the library is automatically a "quality mark", showing other developers that the component has been verified and the risk of problems when using it is significantly reduced.
- Creating proprietary components that can be distributed under a paid license.

ostis-system
⊃ *maternal ostis-system*
:= [ostis-system that includes a library of reusable components]
∋ *OSTIS Metasystem*
⊃ *child ostis-system*
:= [ostis-system that contains a component borrowed from a library of reusable components]

Integration of reusable ostis-systems components is reduced to bonding key nodes by identifiers and eliminating possible duplications and contradictions based on the specification of the component and its content. Integration of any ostis-systems components occurs automatically, without the intervention of the developer. This is achieved through the usage of the *SC-code* and its advantages, the unification of the specifications of reusable components, and the thorough selection of components in libraries by the expert community responsible for this library.

VI. Concept of a reusable ostis-systems component

A reusable ostis-systems component is understood as a component of some ostis-system that can be used within another ostis-system. This is a component of the ostis-system that can be used in other ostis-systems (***child ostis-systems***) and contains all those and only those sc-elements that are necessary for the functioning of the component in the child ostis-system. In other words, it is

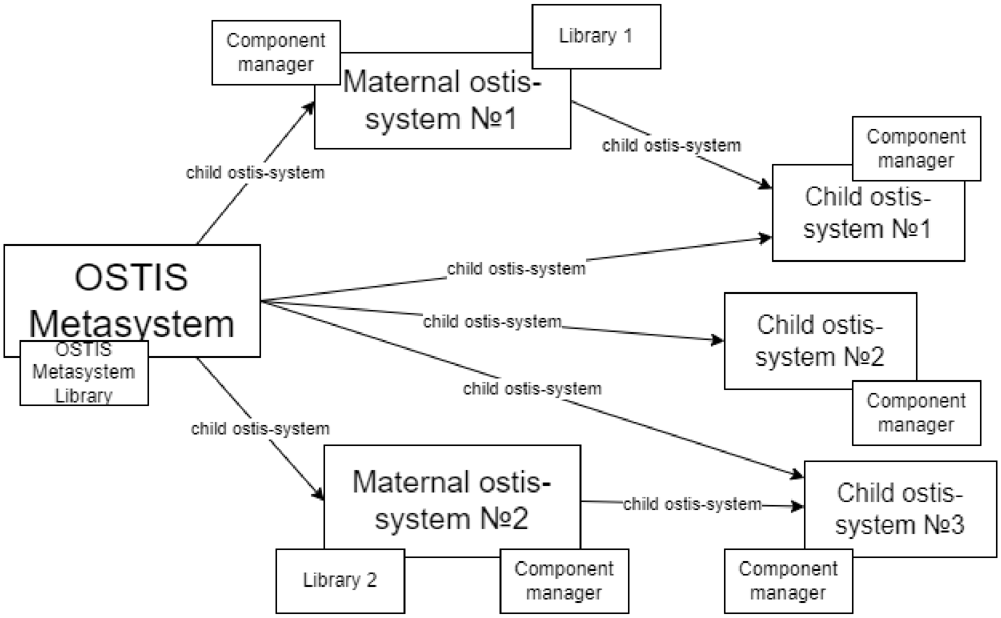


Figure 1. Architecture of the OSTIS Ecosystem

a component of some ***maternal ostis-system***, which can be used in some ***child ostis-system***. Reusable components must have a unified specification and hierarchy to support compatibility with other components. The compatibility of reusable components leads the system to a new quality, to an additional expansion of the set of problems to be solved when integrating components.

reusable ostis-systems component
:= [reusable OSTIS component]
:= *frequently used sc-identifier**:
[reusable component]
⊂ *ostis-system component*

The ostis-system component is an integral part of the ostis-system, which contains all those (and only those) sc-elements that are necessary for its functioning in the ostis-system. The difference between a reusable ostis-systems component and an ostis-system component is that a reusable component has a specification sufficient to install this component in a child ostis-system. The specification is part in the knowledge base of the ***library of reusable components*** for the corresponding maternal ostis-system.

Necessary requirements for reusable ostis-systems components:

- there is a technical possibility to embed a reusable component into a child ostis-system;
- completeness of a reusable component: the component has to fully perform its functions, correspond to its purpose;
- coherence of a reusable component: a component should logically perform only one task from the subject domain for which it is intended. A reusable component has to perform its functions in the most general way so that the range of possible systems in which it can be embedded is the widest;
- compatibility of a reusable component: the component should strive to increase the level of negotiability of the ostis-systems in which it is embedded and have an ability to be integrated automatically into other systems;
- self-sufficiency of the components (or groups of components) of the technology, i.e. their ability to function separately from other components without losing the reasonableness of their usage.

VII. Classification of reusable ostis-systems components

Let us consider the classification of reusable ostis-systems components. The class of a reusable ostis-systems component is an important part of the component specification, which allows better understanding of the purpose and application scope of this component, as well as the class of a reusable component is the most important criterion for searching for components in the ostis-systems library. The main feature of the classification of reusable components is the attribute of the subject domain to which the component belongs. Here the structure can be quite extensive in accordance with the hierarchy of areas of human activity. The following list is not full, it is a short example.

reusable ostis-systems component

⊃ *reusable component of subject domain of medicine*
⊃ *reusable component of subject domain of mathematics*
⊃ *reusable component of subject domain of economics*
⊃ *reusable component of subject domain of art*
⊃ *reusable component of subject domain of computer systems*
⊃ *reusable component of subject domain of plants*

reusable ostis-systems component
⇒ *subdividing**:
{• *reusable knowledge base component*
⊃ *semantic neighborhood*
⊃ *subject domain and ontology*
⊃ *knowledge base*
⊃ *template of a typical ostis-systems component*
∋ *Template for the subject domain description*
∋ *Template for the relation description*
• *reusable problem solver component*
⊃ *atomic knowledge processing agent*
⊃ *knowledge processing program*
• *reusable interface component*
⊃ *reusable user interface component for display*
⊃ *interactive reusable user interface component*
}

For knowledge base components, the most important feature of the classification of reusable components is the type of knowledge used. For the components of the problem solver, there is a problem-solving model, for interface components – the type of interface in accordance with the classification of user interface components.

reusable ostis-systems component
⇒ *subdividing**:
{• *atomic reusable ostis-systems component*
∋ *semantic neighborhood of set*
∋ *sc-agent of set power calculating*
• *non-atomic reusable ostis-systems component*
∋ *abstract sc-agent of logical inference*
∋ *knowledge base of geometry*
}

The typology of the ostis-systems components by atomicity. An atomic reusable ostis-systems component is a component that in the current state of the ostis-systems library is considered as indivisible, that is, does not contain other components in its structure. The belonging of a reusable ostis-systems component to a class of atomic components depends on its specification and on the currently existing components in the library. A non-atomic reusable component in the current state of the ostis-systems library contains other atomic or non-atomic components in its structure and does not depend on its parts. Without any part of the non-atomic component, its purpose restricts. In general, an atomic component can become non-atomic if it is decided to allocate some of its parts as a separate component. All of the above, however, does not mean that even in the case of its platform independence, the atomic component is always stored in sc-memory as a formed sc-structure.

For example, a platform-independent implementation of the sc-agent will always be represented by a set of scp programs but will be an atomic reusable OSTIS component if none of these programs will be of interest as an independent component. In general, a non-atomic component can become atomic if it is decided for some reason to exclude all its parts from consideration as separate components. It should be noted that a non-atomic component does not necessarily have to be composed entirely of other components – it may also include parts that are not independent components. For example, an agent implemented in the SCP language, which is a non-atomic reusable component, may include both scp programs that may (or may not) be reusable components, as well as an agent scp program that does not make sense as a reusable component.

reusable ostis-systems component
⇒ *subdividing**:
{• *dependent reusable ostis-systems component*
∋ *chemistry visualizer*
∋ *subject domain of artificial neural networks*
• *independent reusable ostis-systems component*
∋ *semantic neighborhood of set*
∋ *interface button component*
}

The typology of ostis-systems components by dependency. A dependent reusable ostis-systems component depends on at least one other component of the ostis-systems library, i.e. it cannot be embedded in a child ostis-system without the components on which it depends. The independent component does not depend on any other component of the ostis-systems library.

reusable ostis-systems component

⇒ *subdividing**:
 {• *reusable ostis-systems component stored as external files*
 • *reusable ostis-systems component stored as an sc-structure*
 }

reusable ostis-systems component stored as external files
⇒ *subdividing**:
 {• *reusable ostis-systems component stored as source files*
 • *reusable ostis-systems component stored as compiled files*
 }

The typology of ostis-systems components by their storage method. At this stage of development of the *OSTIS Technology*, it is more convenient to store components in the form of source texts.

reusable ostis-systems component
⇒ *subdividing**:
 {• *platform-dependent reusable ostis-systems component*
  ⊃ *ostis-platform*
  ⊃ *abstract sc-agent that is not implemented in the SCP Language*
 • *platform-independent reusable ostis-systems component*
  ⊃ *reusable knowledge base component*
  ⊃ *SCP agent*
  ⊃ *SCP program*
 }

The typology of ostis-systems components depending on the ostis-platform. A platform-dependent reusable ostis-systems component is a component partially or fully implemented with the help of any third-party means from the point of view of the *OSTIS Technology*. The disadvantage of such components is that the integration of such components into intelligent systems may be accompanied by additional difficulties depending on the specific means of implementing the component. As a potential advantage of platform-dependent reusable ostis-systems components, it is possible to allocate their, as a rule, higher performance due to their implementation at a level closer to the platform. In general, a platform-dependent reusable ostis-systems component can be supplied either as a set of source codes or compiled. The process of integrating a platform-dependent reusable ostis-systems component into a child system developed using the OSTIS Technology strongly depends on the implementation technologies of this component and in each case may consist of various stages. Each platform-dependent reusable ostis-systems component must have the appropriate detailed, correct, and understandable instructions for its installation and implementation in the child system. A platform-independent reusable ostis-systems component is a component that is entirely represented in the *SC-code*. In the case of a non-atomic reusable component, this means that all the simpler components that are part of it must also be platform-independent reusable ostis-systems components. The process of integrating a platform-dependent reusable ostis-systems component into a child system developed using the OSTIS Technology is significantly simplified by using a common unified formal basis for knowledge representation and processing.

The most valuable are platform-independent reusable ostis-systems components.

reusable ostis-systems component
⇒ *subdividing**:
 {• *dynamically installed reusable ostis-systems component*
  := [reusable component, the installation of which does not require a restart of the system]
 • *reusable component, the installation of which requires a restart of the system*
 }

dynamically installed reusable ostis-systems component
⇒ *decomposition**:
 {• *reusable component stored as compiled files*
 • *reusable knowledge base component*
 }

The typology of ostis-systems components according to the dynamics of their installation. The process of integrating components of different types at different stages of the ostis-systems life cycle can be different. The most valuable components are those that can be integrated into a working system without stopping its functioning. Some systems, especially control ones, cannot be stopped, but components need to be installed and updated.

reusable ostis-systems component
⊃ *typical subsystem of ostis-systems*
 ∋ *Environment for the collective development of ostis-systems knowledge bases*
 ∋ *Visual web-oriented editor of sc.g-texts*

For storing reusable ostis-systems components, some storage is required. Such storage can be either an ostis-system or a third-party storage, for example, a cloud service. In addition to the external files of the component, its specification must be located in the storage.

VIII. Specification of reusable ostis-systems components

Each *reusable ostis-systems component* must be specified within the library. The specification includes basic knowledge about the component, which allows ensuring the building of a complete hierarchy of components and their dependencies, and also provides unrestricted integration of components into child ostis-systems. Both relations and component classes are used for component specification.

In order for a reusable component to be accepted into the library, it is required to specify it using a *relation, necessary for installation, that specifies a reusable ostis-systems component*. At the same time, a *relation, optional for installation, that specifies a reusable ostis-systems component* helps to better understand the essence of the component, simplifies the search, but is not necessary for the installation of the component in the ostis-system.

relation specifying a reusable ostis-systems component^

:= [relation that is used in the specification of a reusable ostis-systems component]

⇒ *subdividing**:

{
- *relation, necessary for installation, that specifies the reusable ostis-systems component*
  - ∋ *installation method**
  - ∋ *storage address**
  - ∋ *component dependencies**
- *relation, optional for installation, that specifies a reusable ostis-systems component*
  - ∋ *related component**
  - ∋ *change history**
  - ∋ *authors**
  - ∋ *note**
  - ∋ *explanation**
  - ∋ *identifier**
  - ∋ *key sc-element**
  - ∋ *purpose**

}

The installation method allows the user to install the component manually and the ***component manager*** – automatically. Two main methods of installing reusable components are the method of installing a dynamically installed reusable ostis-systems component and the method of installing a reusable component, when installing which the system requires a restart. With a dynamic installation, it is only necessary to download the component – and it immediately works in the system.

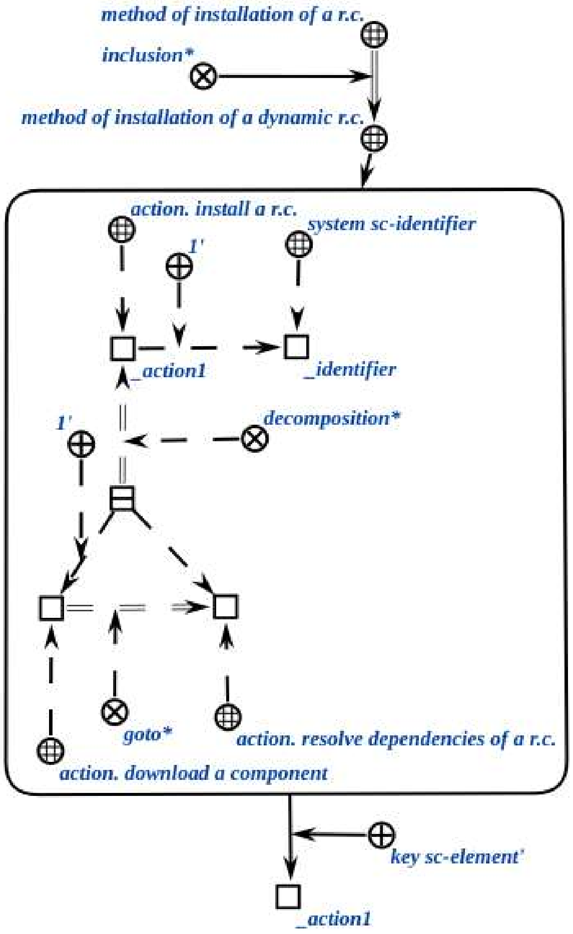

method of installation of a r.c.
inclusion*
method of installation of a dynamic r.c.
action. install a r.c.
system sc-identifier
1'
_action1
_identifier
1'
decomposition*
goto*
action. resolve dependencies of a r.c.
action. download a component
key sc-element'
_action1

Figure 2. The method of installation of a dynamically installed reusable ostis-systems component

When installing a component, during the installation of which the system requires a restart, it is necessary to translate it into the system memory in addition to downloading the component.

The connectives of the *storage address** relation link a reusable component stored as external files and a file containing the URL of a reusable ostis-systems component. Such a file can be a file containing the URL on GitHub of a reusable ostis-systems component, a file containing the URL on Google Drive of a reusable ostis-systems component, a file containing the URL on Docker Hub of a reusable ostis-systems component, and others.

The connectives of the *component dependencies** relation link a reusable component and a set of components, without which the installed component cannot be embedded in a child ostis-system. This components must be successfully installed before the installation of the dependent component.

In some cases, it may turn out that in order to use one reusable OSTIS component, it is advisable or even necessary to additionally use several other reusable OSTIS

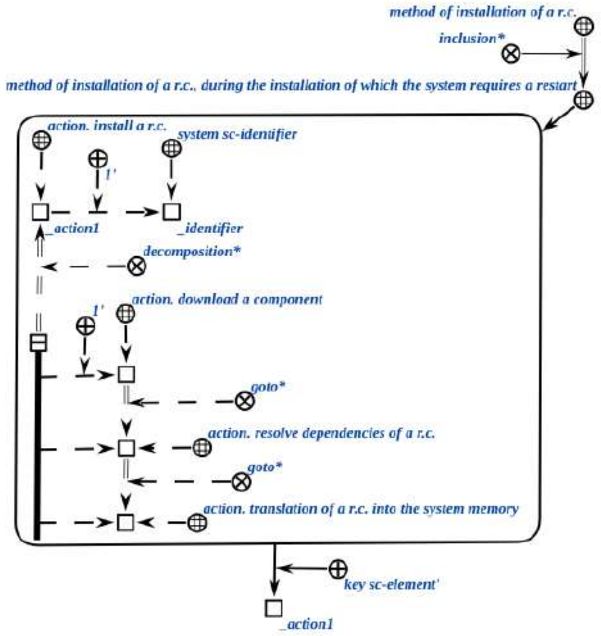

method of installation of a r.c.
inclusion*
method of installation of a r.c., during the installation of which the system requires a restart
action. install a r.c.
system sc-identifier
1'
_action1
_identifier
decomposition*
action. download a component
1'
goto*
action. resolve dependencies of a r.c.
goto*
action. translation of a r.c. into the system memory
key sc-element'
_action1

Figure 3. The method of installing a dynamically installed reusable component, during the installation of which the system requires a restart

components. For example, it may be advisable to use the appropriate interface command combined with any sc-agent of information search, which is represented by a separate component and will allow the user to ask a question for the specified sc-agent through the system interface. In such cases, the related component* relation is used to link components. The presence of such links makes it possible to eliminate possible problems of incomplete knowledge and skills in the child system, due to which any of the components may not perform their functions. The connectives of the related component* relation link reusable ostis-systems components that it is advisable to use in a child system together. Each such connective can additionally be provided with a sc-comment or sc-explanation reflecting the essence of the specified dependency.

IX. Manager of reusable ostis-systems components

The manager of reusable ostis-systems components is a subsystem of the ostis-system, through which interaction with the library of ostis-systems components takes place. The manager of reusable ostis-systems components should be implemented using as few dependencies (ostis-platform components dependencies as well as external dependencies) as possible to provide the maximum level of configurability of developed ostis-sistems.

manager of reusable ostis-systems components
:= *frequently used sc-identifier**:

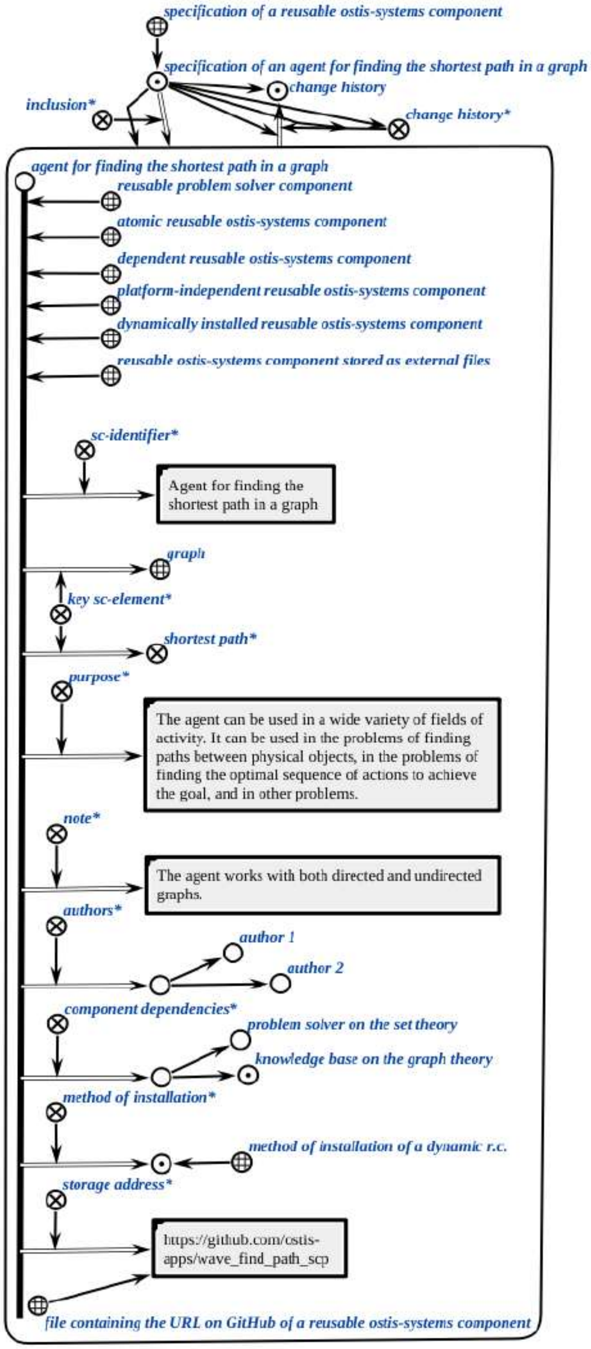

specification of a reusable ostis-systems component
specification of an agent for finding the shortest path in a graph
change history
inclusion*
change history*
agent for finding the shortest path in a graph
reusable problem solver component
atomic reusable ostis-systems component
dependent reusable ostis-systems component
platform-independent reusable ostis-systems component
dynamically installed reusable ostis-systems component
reusable ostis-systems component stored as external files
sc-identifier*
Agent for finding the shortest path in a graph
graph
key sc-element*
shortest path*
purpose*
The agent can be used in a wide variety of fields of activity. It can be used in the problems of finding paths between physical objects, in the problems of finding the optimal sequence of actions to achieve the goal, and in other problems.
note*
The agent works with both directed and undirected graphs.
authors*
author 1
author 2
component dependencies*
problem solver on the set theory
knowledge base on the graph theory
method of installation*
method of installation of a dynamic r.c.
storage address*
https://github.com/ostis-apps/wave_find_path_scp
file containing the URL on GitHub of a reusable ostis-systems component

Figure 4. An example of a specification of a reusable ostis-systems component

[manager of reusable components]
:= *frequently used sc-identifier**:
[manager of components]
⇒ *generalized decomposition**:
{• *knowledge base of the manager of reusable ostis-systems components*
• *problem solver of the manager of reusable ostis-systems components*
• *interface of the manager of reusable ostis-systems components*
}

The knowledge base of the manager of components contains all the knowledge that is necessary to install a reusable component in a child ostis-system. Such knowledge includes knowledge about the specification of reusable components, methods of installing components, knowledge about the ostis-systems libraries with which interaction takes place. The problem solver of the manager of components interacts with the ostis-systems library and allows installing and integrating reusable components into a child ostis-system, as well as searching, updating, publishing, and deleting components. The interface of the manager of reusable components provides convenient usage of the manager of components for the user and other systems.

The functionality of the manager of ostis-systems components is as follows:

- **Search for reusable ostis-systems components**. The set of possible search criteria corresponds to the specification of reusable components. Such criteria can be component classes, its authors, identifier, fragment of a note, purpose, belonging to a subject domain, type of component knowledge, and others.
- **Installation of a reusable ostis-systems component**. The installation of a reusable component takes place regardless of the typology, installation method, and location of the component. A necessary condition for the possibility of installing a reusable component is the availability of the ***specification of a reusable ostis-systems component***. Before installing a reusable component in a child system, it is necessary to resolve all dependencies by installing dependent components. After successful installation of the component, an information construction is generated in the knowledge base of the child system, indicating the fact of installing the component into the system using the *installed components** relation. After installing the component in the ostis-system, contradictions may arise in the knowledge base, which are eliminated by means of detecting and analyzing errors and contradictions in the ostis-system knowledge base.
- **Publishing a reusable ostis-systems component to the ostis-systems library**. When a component is published to the ostis-systems library, verification takes place based on the component specification. It is also possible to update the version of the published component by the community of its developers.
- **Updating an installed reusable ostis-systems component**.
- **Deleting an installed reusable component**. As in the case of installation, after deleting a reusable component from the ostis-system, the fact of deleting the component is established in the knowledge base of the system. This information is an important part of the operational history of the ostis-system.
- **Adding and deleting libraries tracked by the ostis-system**. The manager of components contains information about a variety of sources for installing components, the list of which can be supplemented manually. By default, the manager of components tracks the OSTIS Metasystem Library, however, it is possible to create and add extra ostis-systems libraries.

X. Conclusion

In the article, the implementation of a library of reusable compatible components of intelligent computer systems based on the OSTIS Technology is proposed, which makes it possible to use a component-based approach to the design of intelligent systems and reduce the time and complexity of system development, as well as increase the level of compatibility of systems using reusable ostis-systems components.

The classification and specification of reusable ostis-systems components are clarified, the concepts of a components library and manager are considered.

An example for the specification of a component that can be found in the library of compatible ostis-systems components is given. The architecture of the ecosystem of intelligent computer systems is considered from the point of view of using a library of reusable components.

The results obtained will improve the design efficiency of intelligent systems and automation tools for the development of such systems, as well as provide an opportunity not only for the developer but also for the intelligent system to automatically supplement the system with new knowledge and skills.

References

[1] Natalia N. Skeeter, Natalia V. Ketko, Aleksey B. Simonov, Aleksey G. Gagarin, Irina Tislenkova, “Artificial intelligence: Problems and prospects of development,” *Artificial Intelligence: Anthropogenic Nature vs. Social Origin*, 2020.

[2] Olena Yara, Anatoliy Brazheyev, Liudmyla Golovko, Liudmyla Golovko, Viktoriia Bashkatova, “Legal regulation of the use of artificial intelligence: Problems and development prospects,” *European Journal of Sustainable Development*, 2021.

[3] Golenkov, V. V., “Methodological problems of the current state of works in the field of artificial intelligence,” *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*, pp. 17–24, 2021.

[4] Iyengar, Ashvin, *Component Design for Relational Databases*, 12 2021, pp. 143–156.

[5] Ford, Brian and Schiano-Phan, Rosa and Vallejo, Juan, *Component Design*, 11 2019, pp. 160–174.

[6] Donatis, Antonio, *OOP in Component Design*, 01 2006.

[7] Nazia Bibi, Tauseef Rana , Ayesha Maqbool, Tamim Alkhalifah, Wazir Zada Khan, Ali Kashif Bashir, Yousaf Bin Zikria, "Reusable component retrieval: A semantic search approach for low resource languages," *ACM Transactions on Asian and Low-Resource Language Information Processing*, 2022.

[8] Bukhari, S. A. C., Krauthammer, M. & Baker, C. J. O., "An architecture for biomedical image discovery, interoperability and reusability based on semantic enrichment," *SWAT4LS(Citeseer, 2014)*, 2014.

[9] Ryndin, Nikita and Sapegin, Sergey, "Component design of the complex software systems, based on solutions' multivariant synthesis," *International Journal of Engineering Trends and Technology*, vol. 69, pp. 280–286, 12 2021.

[10] L. Cafaro, R. Francese, C. Palumbo, M. Risi, and G. Tortora, "An agile process supporting software reuse: An industrial experience," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1544–1551.

[11] Shunkevich D.V., Davydenko I.T., Koronchik D.N., Zukov I.I., Parkalov A.V., "Support tools knowledge-based systems component design," *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, pp. 79–88, 2015. [Online]. Available: http://proc.ostis.net/proc/Proceedings%20OSTIS-2015.pdf

[12] X. Qu, X. Feng, Y. Zhang, S. Wang, L. Sun, P. Hua, and Y. Wang, "Research on component retrieval and matching methods," in *2022 International Seminar on Computer Science and Engineering Technology (SCSET)*, 2022, pp. 358–362.

[13] T. Diamantopoulos and A. L. Symeonidis, "Mining source code for component reuse," in *Mining Software Engineering Data for Software Reuse*. Springer, 2020, pp. 133–174.

[14] Blähser, Jannik and Göller, Tim and Böhmer, Matthias, "Thine — approach for a fault tolerant distributed packet manager based on hypercore protocol," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2021, pp. 1778–1782.

[15] V. A. Buregio, E. S. Almeida, D. Lucredio, and S. L. Meira, "Specification, design and implementation of a reuse repository," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, vol. 1, 2007, pp. 579–582.

[16] Fritzson, Peter, *Modelica Library Overview*, 2015, pp. 909–975.

[17] Prakash Pradhan, Sanjaya, *Working with Microsoft Power Apps*. Berkeley, CA: Apress, 2022, pp. 79–131. [Online]. Available: https://doi.org/10.1007/978-1-4842-8600-5_3

[18] Memduhoğlu, Abdulkadir and Basaraner, Melih, "Possible contributions of spatial semantic methods and technologies to multi-representation spatial database paradigm," *International Journal of Engineering and Geosciences*, vol. 3, pp. 108–118, 10 2018.

[19] M. Atzeni and M. Atzori, "Codeontology: Rdf-ization of source code," in *International Semantic Web Conference*. Springer, 2017, pp. 20–28.

[20] B. Antunes, P. Gomes, and N. Seco, "Srs: A software reuse system based on the semantic web," in *3rd International Workshop on Semantic Web Enabled Software Engineering (SWESE)*, 2007.

[21] V. Gribova,L. Fedorischev, P. Moskalenko, V. Timchenko, "Interaction of cloud services with external software and its implementation on the IACPaaS platform," pp. 1–11, 2021.

[22] Vladimir Golenkov and Natalia Guliakina and Daniil Shunkevich, *Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[23] (2022, September) IMS.ostis Metasystem. [Online]. Available: https://ims.ostis.net

Комплексная библиотека многократно используемых семантически совместимых компонентов интеллектуальных компьютерных систем нового поколения

Орлов М.К.

Важнейшим этапом эволюции любой технологии является переход к компонентному проектированию на основе постоянно пополняемый библиотеки многократно используемых компонентов. В работе рассматривается подход к проектированию систем, управляемых знаниями, ориентированный на использование совместимых многократно используемых компонентов, что существенно сокращает трудоемкость разработки таких систем.

Received 30.10.2022

Methods and tools for designing and analyzing the quality of knowledge bases of next-generation intelligent computer systems

Stanislau Butrin
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: stas.butrin1331@gmail.com

***Abstract*—In the article, an ontological approach to the design of knowledge bases of next-generation intelligent computer systems is considered. It is based on the usage of a multi-agent approach to ensure the consistency of tools for knowledge base quality analysis. The results will improve the efficiency of designing knowledge bases of intelligent computer systems.**

***Keywords*—knowledge base, ontology, subject domain, knowledge base verification, knowledge base quality analysis**

I. Introduction

An important phase in the development of any system is quality control, since the degree of liveness and effectiveness of the system are determined at this stage.

It is believed that the quality of intelligent computer systems is largely determined by the quality of their knowledge bases [1].

Currently, tools for creating knowledge bases of intelligent computer systems are rapidly developing. However, the development of knowledge bases is a collaborative process, which is characterized by the occurence of contradictions and misunderstandings. Therefore, special attention should be paid to the means of checking and verification of knowledge operated by intelligent systems.

In the real world, contradictions and errors are inevitable, since humans, like the systems created by them, will be limited to a certain picture of the world, within which some knowledge will be considered as true and others as false. Nevertheless, the system must be able to adapt, adjust its picture of the world and its concept of certain information, in order to effectively perform the tasks for which it was created.

II. Analysis of existing approaches to solving the problem

Verification is a type of quality analysis and part of the development process. It consists in checking the information for correctness and accuracy. Its purpose is to identify errors, various defects, and shortcomings in order to eliminate them in time.

Currently existing verification methods are well-developed, as well as a large number of different verification models using extended decision tables, Petri nets [2], various logics, such as logics with vector semantics [3], [4], and other models. Moreover, specialized ontologies are formed to describe a variety of means and models of knowledge base verification [5]. However, there is no mechanism for interaction of the tools using these methods.

Most of the work in the field of verification focus on a particular approach or model, although the most effective approach to verification is a combination of different methods.

Therefore, knowledge base verification tools that currently exist have a number of problems such as [6]:

- dependence on the format of information representation, because of what it is necessary to spend additional time on converting information;
- the problem of impossibility to be reused, since the tools are usually created taking into account the specifics of a particular system;
- problem of lack of mechanism for interaction between means of verification and knowledge analysis;
- high role of the human in the verification process, because the most common way of verifying databases is a manual check of the database by an expert; a human acts as an administrator, making an unanimous decision, imposing their opinion on the system;
- modern tools do not take into account and do not consider the process of verification within the interaction of systems with each other.

These problems could be solved if:

- developers used a unified and convenient knowledge representation format;
- systems were created using a common methodology and were compatible with each other;
- experts thought over and implemented a mechanism that allows the system to try to make a decision

about its state and presence of problems and errors; the system may not always make the right decisions, but those should be its mistakes, not those of experts and developers.

Having analyzed the works carried out in this area, it is possible to notice a decline of interest in the verification of knowledge bases. A possible reason for this is that in the absence of a unified methodology for the development of intelligent computer systems and their knowledge bases, it is inappropriate to develop a methodology for the design of tools for knowledge base quality analysis.

Thus, at the moment, the problem of complex analysis and verification is relevant because of the lack of methodology for the design of analysis and verification tools that can effectively interact with each other to solve the problem.

III. Proposed approach

Within this article, an OSTIS Technology is used as the proposed approach. This technology is a complex of models, tools, and methods designed for the development of next-generation intelligent computer systems.

The advantages of the OSTIS Technology within the verification problem are:

- availability of a common methodology for the design of intelligent systems, which allows solving the problem of compatibility of systems during their interaction;
- all knowledge is represented in a unified form, which allows effectively processing them, reducing the cost of converting to a minimum;
- means by which contradictions are detected, analyzed, and resolved are described in the knowledge base, and their specification is represented in the system knowledge base, thereby making it easy to expand them and let the system know what tools it contains;
- absence of semantic equivalent fragments, which ensures that corrections are made locally and eliminates the need to make corrections repeatedly in different places;
- multi-agent approach, which allows considering means of analysis and verification of knowledge bases as a collective of agents, capable of interacting with each other and then making a joint decision about the state of the knowledge base.

Within the OSTIS Technology, works related to verification have already been conducted [7] but did not touch on the subject of verification in sufficient detail, in particular, there is no description of the approach to the design of verification means and a mechanism that would ensure their compatibility, while other works considered more special cases, such as verification during knowledge integration [8]. However, the verification of the knowledge base and intelligent system is not limited to this.

The OSTIS Technology uses subject domains to formalize knowledge, allowing allocating only a certain class of entities under study from the diversity of the World, focusing attention only on something specific. Ontologies are used to specify subject domains. By ontology, the semantic specification of any knowledge, which has a rather complex structure, is meant.

The *OSTIS* Technology is based on the usage of unified semantic networks with a basic set-theoretic interpretation of their elements as a method of knowledge representation. This way of knowledge representation is called an *SC-code*, and the semantic networks, represented in the *SC-code*, are called *sc-graphs* (*sc-texts*, or *texts of the SC-code*). The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, can be *sc-arcs* or *sc-edges* depending on their orientation). The *Alphabet of the SC-code* consists of five basic elements, on the basis of which SC-code constructions of any complexity are built, including the introduction of more particular kinds of sc-elements (e.g., new concepts). The memory storing SC-code constructions is called semantic memory, or *sc-memory*.

The technology also offers several universal options for visualizing *SC-code* constructions, such as *SCg-code* (graphical variant), *SCn-code* (nonlinear hypertext variant), *SCs-code* (linear string variant).

The OSTIS Technology uses a multi-agent approach, which allows conveniently solving the problem of interaction of verification means, since in this case, the verification means should be considered as a collective of agents.

Thus, the proposed approach comes down to the development of:

- specialized *subject domain and ontology*, which would contain all the necessary knowledge about the possible types of problem fragments of the knowledge base and ways to fix them;
- an algorithm that would allow the system to identify problem fragments in itself and eliminate them, while ensuring the consistency of the means of the system;
- a specialized problem solver, containing the necessary agents to identify and eliminate the problem fragments.

IV. Analysis of knowledge base quality

The quality of the knowledge base is largely determined by the level of presence/absence of non-factors [9] in the knowledge base.

non-factor
:= [group of semantic properties that determine the quality of information stored in the memory of a cybernetic system]
= {

- *correctness/incorrectness of the information stored in the memory of a cybernetic system*
- *uniqueness/uniqueness of the information stored in the memory of a cybernetic system*
- *integrity/unintegrity of information stored in the memory of a cybernetic system*
- *compliance/incompliance of information stored in the memory of a cybernetic system*
- *reliability/unreliability of information stored in the memory of a cybernetic system*
- *accuracy/inaccuracy of information stored in the memory of a cybernetic system*
- *certainty/uncertainty of information stored in the memory of a cybernetic system*
- *determinacy/undeterminacy of information stored in the memory of a cybernetic system*

}

In this article, the focus is on such non-factors as:consistency, incompleteness, incompliance.

consistency/inconsistency of the information stored in the memory of a cybernetic system

:= [level of presence of various kinds of contradictions and, in particular, errors in the stored information]

contradiction*

:= [pair of contradictory fragments of information stored in the memory of a cybernetic system*]

⇒ *note**:

[The most common contradictory fragments of information are:

□ some regularity (rule), explicitly represented in memory

□ information fragment that does not correspond (contradict) to this regularity

]

completeness/incompleteness of information stored in the memory of a cybernetic system

:= [extent to which the information stored in the memory of a cybernetic system describes that system environment of existence and the problem-solving methods it uses (in sufficient detail) for the cybernetic system to actually be able to solve all the set of problems corresponding to it]

compliance/incompliance of information stored in the memory of a cybernetic system

:= [variety of forms and total amount of information garbage included in the information stored in the memory of a cybernetic system]

The process of creating and editing the ostis-system knowledge base is reduced to the formation of proposals by developers to edit a particular section of the knowledge base. Subsequently, these proposals are considered by the administrators of the knowledge base. The scheme of the knowledge base with proposals is shown in Figure 1.

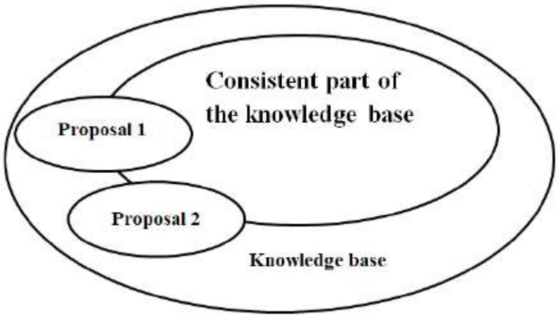


Figure 1. A knowledge base scheme with proposals made

The main reason of increasing the level of non-factors in the knowledge base is the occurence of new information or changes in existing one.

Examples of such changes are cases where:

- the user creates or modifies a fragment of the knowledge base;
- the system obtains new information by merging knowledge bases or by using tools to automatically provide the knowledge base with data from various sources;
- changes occur in the system during the work of the agents.

In the proposed approach, the structures in the knowledge base, which increase the level of non-factors in the system, must be localized and described, so that they can be fixed in the future. It is important that the mechanism of localization and description itself should be universal. This implies that the processing of such structures should not depend on their type.

V. Subject domain of problem structures

Therefore, there is a need to develop a specialized Subject domain, which would describe this kind of structures. For this purpose, the concept of a problem structure is introduced.

problem structure

:= [structure describing a unsatisfactory knowledge base fragment]

⇐ *combination**:

{● *incorrect structure*

:= [structure containing fragments that contradict any rules or patterns described in the knowledge base]

- *structure describing incompleteness in the knowledge base*

  := [structure in which there is incompleteness (i.e., there are a number of information holes)]

  ⇒ *note**:

  [A structure describing an incompleteness in the knowledge base is a structure containing a fragment of the knowledge base in which some information is missing that is necessary (or at least desirable) for an unambiguous and complete understanding of the meaning of the fragment.]

- *information garbage*

  := [structure whose removal would not significantly complicate the system]

  := [structure containing a fragment of the knowledge base that, for whatever reason, has become unnecessary and requires deletion]

}

In addition to the problem structure itself, the Subject domain addresses its more special cases and related concepts.

incorrect structure

⇐ *inclusion**:

{• *duplication of system identifiers*
- *mismatch of connective elements to relation domains*
- *cycle in relation order*
- *structure that contradicts the singularity property*

}

structure describing incompleteness in the knowledge base

⇐ *inclusion**:

{• *no maximum class of research subjects of the subject domain specified*
- *for the entity, the system identifier is specified, but no main identifiers are specified for all external languages*
- *no relation domains specified*
- *concept is not associated with any subject domain*

}

VI. Algorithm for knowledge base fragment verification

To ensure the compatibility of verification tools it is required to develop an algorithm that allows eliminating problem structures in the knowledge base of an intelligent system in a unified way.

The process of verification and correction of the structure in this algorithm should be considered as an iterative process, in which, after proposing any changes, the following should be checked:

- if it has ceased to be problematic;
- whether changes have created new problem structures.

If it is not possible to propose changes to fix it, the structure must be reverted to its original state.

Taking into account the features mentioned above, the general algorithm for working with problem structures in the knowledge base should include the following steps:

- identifying problem structures;
- fixing the state of the problem structure;
- proposing changes to correct the problem structure;
- applying the proposed changes;
- checking the changed structure;
- rolling back in case of impossibility to correct the structure;
- fixing the non-corrected structure.

VII. Problem solver of knowledge base verification

The tools for quality analysis within the OSTIS Technology are agents. At a minimum, agent interactions should include the ability for an agent to initiate other agents and to access the results of their work.

The organization of the mechanism for interaction of the corresponding means of the verification process can be carried out by the corresponding agents. An example of such a system is a system in which an agent monitors the state of a knowledge base and, in the case of new information arrival, initiates the appropriate verification means. The initiated means will analyze the received information and record its state in the base. If necessary, the means of correction will propose the appropriate changes and apply them.

The system of such verification tools will be a problem solver, an example of the possible structure of which is represented below.

Problem solver of means for identifying and eliminating contradictions

⇐ *decomposition of an abstract sc-agent**:

{• *non-atomic agent for contradiction detection*

  := [Set of agents providing contradiction retrieval and fixation in the structure]

- *Non-atomic agent for contradiction elimination*
  := [Set of agents creating proposals to fix contradictions]
- *Agent applying proposals to fix contradictions*
- *Non-atomic agent for structure verification*

}

VIII. Version control model

For the coordinated interaction of agents of an intelligent system, only their specifications are not enough, but also a format for describing the structures with which they will work is required. Such a format is a version control model, which must take into account the changes made to the structure in the process of fixing it, while being convenient for the work of the agents.

It is supposed that correction agents do not apply the changes immediately but only make proposals for changes.

Thus, the corrected structure is only a modified version of the original structure.

This is done, among other things, to avoid providing the knowledge base with copies of the initial structure.

An example of a version control model [10] that can be used is shown in Figure 2.

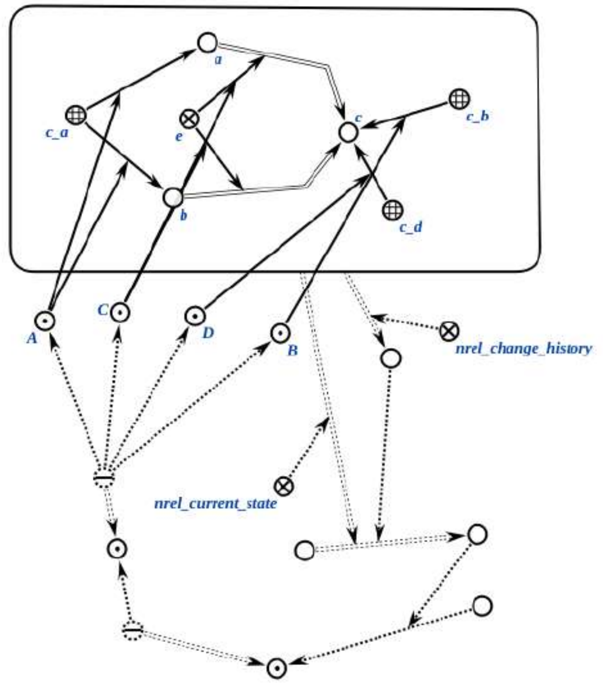


Figure 2. An example of versioning model

IX. Description of problem solver agents

Verification agents can be divided into:

- agents detecting problem structures;
- agents correcting/removing problem structures.

The stage of problem structure detection implies the initiation of agents for problem structure detection. This can be carried out by some agent which has knowledge of what search agents are in the system, or agents can be initiated, for example, when new information is added to the knowledge base. The problem of agents for problem structure detection is to find in the knowledge base the fragments causing problems, describe, and record information about them, so that later the correction agents could make appropriate changes to them. The result of the work of such agents in the general case is the immersion of the problem fragment in the structure belonging to the corresponding classes of problem structures. An example of the result of the agent for problem structure detection is shown in Figure 3.

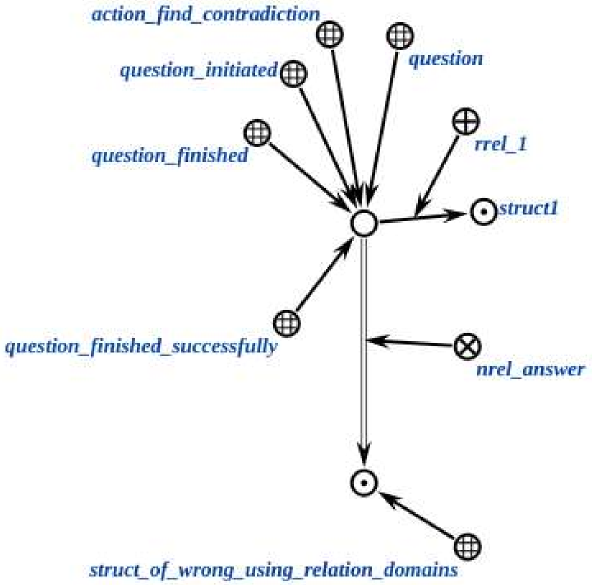

action_find_contradiction
question_initiated
question
question_finished
rrel_1
struct1
question_finished_successfully
nrel_answer
struct_of_wrong_using_relation_domains

Figure 3. An example of the result of the agent for problem structure detection

The stage of fixing the state of the problem structure implies the usage of the version control model to fix the state of the structure. For the initial state, the elements belonging to the structure must be marked.

The stage of proposing changes to fix the structure involves the work of the agents forming proposals to change the structure. These agents can be called either by the supra-agent-coordinator on the basis of what kinds of problem structures they can fix, or they can respond to an event themselves, for example, on adding the belonging of the structure to the appropriate class of problem structures. The result of the work of such agents is sets of elements that should be removed or added to the structure so that it ceases to be problematic.

An example of the result of the agent for structure change proposal is shown in Figure 4

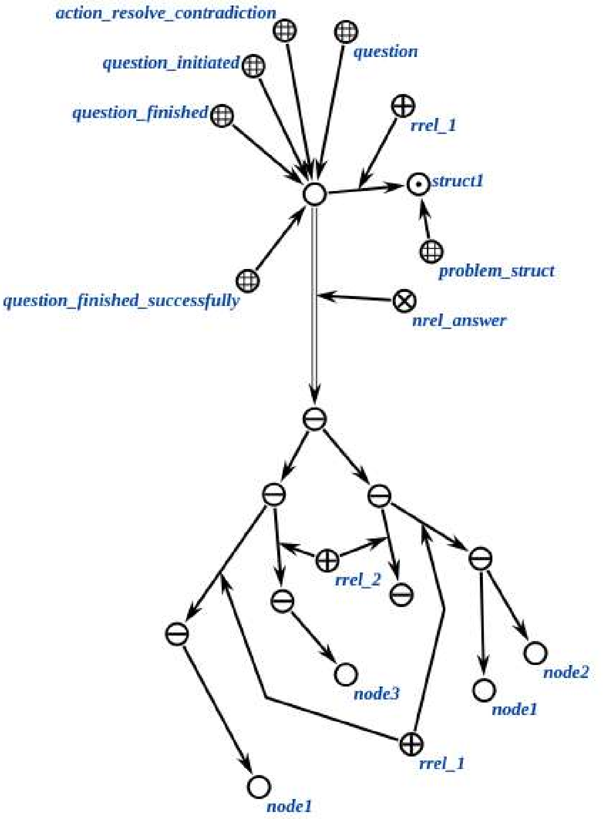


Figure 4. An example of the result of the agent for structure change proposal

Further, the stage of applying the changes proposed by the agents takes place, after which the state of the knowledge base is checked. This is necessary to make sure that:

- structure is fixed and is no longer problematic;
- fixes over the problem structure have not generated new problem structures that the system cannot fix.

In the case where the system is unable to propose changes capable of fixing the problem fragment, the structure should be returned to its original state. It is important to fix the fact that the system at the moment is not able to fix the problem structure on its own. This is necessary to avoid further unnecessary attempts to fix this structure, as well as to indicate that the solution to this problem may require the help of an expert or developer.

X. Conclusion

In the article, an approach to the design of tools for analyzing the quality of knowledge bases of next-generation intelligent computer systems is proposed. It is based on the usage of a multi-agent approach to ensure the consistency of tools for knowledge base quality analysis.

The subject domain of problem structures is allocated, as well as the algorithm of interaction between agents for verification of knowledge bases, which allows describing the problem fragments of the knowledge base and designing tools that can consistently analyze and improve the quality of the knowledge base.

The obtained results allow increasing the efficiency of the development of tools for analyzing the quality of knowledge bases, in particular, the means of verification, which ultimately allows improving the quality of the knowledge bases themselves.

References

[1] C. Gavrilova, *Gavrilova T.A. Knowledge Bases of Intelligent Systems / T.A. Gavrilova, V.F. Khoroshevsky. - SPb: Peter, 2000*, 2000.

[2] L. Martin and A. Romanovsky, “Stochastic activity networks for the verification of knowledge bases,” 08 2017, pp. 37–44.

[3] L. Arshinskiy, A. Ermakov, and M. Nitezhuk, “Logic with vector semantic as a means of knowledge bases verification,” *Ontology of designing*, vol. 9, pp. 111–122, 12 2019.

[4] ——, “Complex verification of rule-based knowledge bases using VTF-logic,” *Ontology of designing*, vol. 10, pp. 112–120, 04 2020.

[5] G. Rybina and V. Smirnov, “Methods and algorithms of knowledge base verification in integrated expert systems / G.V. Rybina, V.V. Smirnov // Izvestia RAN. Theory and Control Systems 2007,” vol. 4, pp. 91–102, 11 2005.

[6] D. Zhang, “Knowledge base verification: issues and approaches,” 10 2022.

[7] I. Davydenko, “Semantic models, method and tools of knowledge bases coordinated development based on reusable components,” in *Open semantic technologies for intelligent systems*, V. Golenkov, Ed., BSUIR. Minsk , BSUIR, 2018, pp. 99–118.

[8] V. Ivashenko, “Modeli i algoritmy integratsii znanii na osnove odnorodnykh semanticheskikh setei [Models and algorithms of knowledge integration based on homogeneous semantic networks],” avtoref. dis… kand. tekhn. nauk: 05.13.17, V.P. Ivashenko ; Uchrezhdenie obrazovaniya «Belorusskii gosudarstvennyi universitet informatiki i radioelektroniki», Minsk, 2014.

[9] A. Narin’jani, “Ne-faktory: kratkoe vvedenie [Non-factors: a brief introduction],” *Novosti iskusstvennogo intellekta [Artificial intelligence news]*, no. 2, pp. 52–63, 2004.

[10] N. Zotov and K. Bantsevich, “Principi obespechenya versionnosti fragmentov baz znaniy intellectalnich sistem [Principles of providing versioning of fragments of knowledge bases of intelligent systems],” *Information technology and management : proceedings of the 58th scientific conference of graduate students, undergraduates and students*, p. 69, 2022.

Методика и средства проектирования и анализа качества баз знаний интеллектуальных компьютерных систем нового поколения

Бутрин С.В.

В работе рассмотрен подход к проектированию средств анализа качества баз знаний интеллектуальных компьютерных систем нового поколения. Он основан на использовании многоагентного подхода для обеспечения согласованности средств анализа качества баз знаний.

Полученные результаты позволяют повысить эффективность разработки средств анализа качества баз знаний, в частности, средств верификации, что в конечном итоге позволяет повысить качество самих баз знаний.

Received 01.11.2022

Methodology and tools for component interface design of next-generation intelligent computer systems

Mikhail Sadouski, Alexandra Zhmyrko
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: sadovski@bsuir.by, aleksashazh@gmail.com

***Abstract*—In the article, the methodology of designing interfaces for next-generation computer systems is considered. The stages of designing adaptive intelligent multimodal user interfaces and the usage of these stages in the context of the OSTIS Technology are described.**

***Keywords*—interface design methodology, adaptive intelligent multimodal user interface, OSTIS, ostis-system interface, next-generation intelligent computer systems**

I. Introduction

Interface design is one of the most important stages in the development of any system.

Design of the interface is often thought of as art rather than science and suffers from lack of formalisms, models, tools, and methodical design approaches. Slowly, the design process is becoming more structured, and more formal tools are becoming available [1].

The user interface is a set of software and hardware tools that ensures the interaction between the user and the system.

The user, when dealing with the interface, must imagine what information about the problem he has and in what state are the means by which they will solve this problem. The effectiveness of the user and their interest is ensured by a properly formulated methodology of development and design of the user interface.

Currently, the organization of user interaction with the computer system is the paradigm of a **competent user**, who knows how to manage the system and is fully responsible for the quality of interaction with it. The variety of forms and types of interfaces leads to the need for the user to adapt to each specific system, to learn the principles of interaction with it for solving the problems they need.

Friendliness of the user interface should consist in the adaptability of the system to the characteristics and qualifications of the user, the exclusion of any problems for the user in the dialog with the intelligent computer system, in a permanent care to improve the communication skills of the user. Consequently, it is necessary to move away from the usual user adaptation to the system (by learning to use it) in the direction of adapting the interface itself to the purposes, problems, and characteristics of a particular user in real time [2].

The interface of next-generation intelligent computer systems should provide interaction with the user on an equal basis, be able to adapt to its characteristics, as well as to perceive different types of information input. The terms of adaptive, intelligent, and multimodal interface are often used to organize such interaction.

The interfaces to be designed must comply with the following aspects:

- Uniformity is one of the main principles of user interface design. In the modern world, users are familiar with using different systems. Comparing systems with each other, it is possible to notice some similarities in their design (for example: search is located at the top of the page, navigation menu is located on the left, etc.). When using the system, the user develops a pattern of thinking.
  A pattern of thinking is knowledge about the system and how it works. When a user operates a system that is new to them, they apply this model to new situations. Accordingly, using a new system provides a lower cognitive load, i.e., users spend less time acquiring the interface. In this case, users can spend more energy to achieving their purposes.
  This suggests that when designing user interfaces, it is necessary to consider general interface design rules that build around existing patterns of thinking, without first having to learn the specifics of how the system works. Designing an interface that meets expectations allows users to apply their knowledge based on previous experience, and some similarity of the new system to the old system allows them to focus on the things that are important to them.
- Ease of usage – interaction with the interface and movement through it should be easy and simple for users, i.e. requiring minimal effort. The time

required for a user to move to and interact with an interactive interface element is a critical parameter. It is important to properly set the size and position of interactive interface elements so that they are easy to find and so that the clickable area for selecting the element meets the user expectations. Currently, there are various ways of selecting elements, such as mouse, finger, stylus, etc. Such elements have different accuracy, which consequently complicates the design of interfaces.

- Simplification. Simplifying an interface or process helps reduce the cognitive load on users and increases the possibility that they will complete their task and achieve their purpose. But it is also worth considering that simplification can affect the user experience negatively – when we simplify everything to the point of meaninglessness and it is no longer clear, what interface actions are available, what the next steps might be, and where to find the correct information.

By adhering to these aspects, it is possible to minimize the complexity of the interaction between users and systems.

At the moment, the following problems in the design of user interfaces are relevant:

- the lack of common methods and tools for designing user interfaces limits the reusage of already developed components and increases the time required to teach the user new user interfaces, which also increases the development time and cost of designing and maintaining user interfaces;
- the extensibility of the interface components is not supported;
- the ability to transfer user interfaces from one implementation platform to another is difficult;
- most systems do not allow modifying the user interface during operation;
- the tools of helping the user to interact with the system interface are usually designed separately from the design of the interface itself;
- interface design tools and the system for which it is intended, as a rule, differ significantly, making it difficult to integrate the interface into the system;
- most systems do not have an ability to flexibly adapt the user interface to the needs of a particular user.

To solve these problems, in the article, an approach to designing user interfaces based on a unified logical-semantic model is proposed.

The design of user interfaces includes a number of sequential stages. Within this article, the design stages of traditional user interfaces and the design stages of adaptive intelligent multimodal user interfaces will be considered.

II. State of art

Adaptive user interface is a set of software and hardware tools that allows the user to use the system most effectively by automatically adjusting the interface to the specific user with respect to their needs and context [3].

Configuration of functionality and interface parameters can be performed either manually by the user or automatically by the system based on the available information about the user. Thus, it is necessary to distinguish between adaptive and adaptable systems – these terms are not synonymous, although in the literature, it is often possible to find a substitution of these concepts [4].

In adaptable systems, any adaptation is predefined and can be changed by users before the system runs. In contrast, in adaptive systems, any adaptation is dynamic, that is, it occurs at the same time as the user interacts with the system, and depends on the user behavior. However, the system can also be adaptable and adaptive at the same time [5].

In the literature, it is also possible to find the term ’adapted interface’. Adapted user interfaces are user interfaces adapted to the end-user at designing time, with no adaptation changes occurring in running time [6].

An Intelligent User Interface (IUI) is a user interface that can assume further user actions and provide information based on that assumption [7].

Next, we consider the stages of designing adaptive intelligent multimodal user interfaces.

A. Design methodology for adaptive intelligent multimodal user interfaces

The author of [8] identifies 4 main stages of design.

The first step is the **Analysis of users, system, and environment**.

The analysis phase is probably the most important phase in any design process, but even more so in IUI design. In the design process of a normal non-intelligent interface it is necessary to analyze who is the average user, what problems the interface should support, and on what system they will be performed. With an IUI there often is no average user. Ideally, an IUI should be able to adapt to any user in any environment. Therefore, the used adaptation technique should be designed in such a way that it can support all types of users. In practice, this is hard to achieve so we simply focus on certain user types. David Benyon [9] has identified five interrelated analysis activities for designing adaptive systems:

- Functional analysis: what are the main functions of the system?
- Data analysis: what is the meaning and structure of data in the application?
- Problem knowledge analysis: which cognitive capabilities do the users need to have, for example, the assumed mental model, known search strategies,

level of cognitive loading, etc? This analysis does require some design to have been completed before it can be performed.
- User analysis: what types of users are there and what are their capabilities, intelligence, and cognitive abilities?
- Environment analysis: in which environment is the system to operate?

The result of the analysis phase is a specification of the users purposes and information they need, as well as the functions and information that is required by the system. A problem that is often encountered in the analysis process of IUIs is the 'paradox of change'. Since there are hardly any common, functional IUIs, it is difficult to analyze how users will interact with them. On the other hand, if these interfaces are developed and become widely used, there is the risk that those systems will influence the analysis process. Wizard of Oz studies can be carried out to overcome this problem. In this kind of study, data is collected from a user who is led to believe that they are working on a fully functional and automatic system while, in fact, the system is being controlled by another human.

The second step is **development and implementation**.

The process of developing new interaction techniques and metaphors is mainly one of creativity. The best way is just to go out and try new concepts and ideas. Of course, there are many general guidelines for interface design that it is necessary to keep in mind. [10] Unfortunately, most of these guidelines were developed for DM-interfaces and are difficult to apply to IUIs. Some DM guidelines, such as consistency and user control, are violated by IUIs. This is also the reason that many DM-interface designers heavily criticized some IUIs concepts. On the other hand, other guidelines are better served by IUIs than by DM interfaces. For example, using natural language, IUIs can 'speak the user's language' much better than DM systems. Also, many IUIs try to reduce the short-term memory load of users by taking over problems. The result of the development and implementation process is a user interface that has a 'look-and-feel' that the designer thinks will suit the users and fulfill the requirements of the analysis phase.

The third step is **evaluation**.

In the evaluation stage of the design process we return to the questions of the analysis phase. The requirements that were drawn up in the analysis phase should be met, and the effectiveness of the prototype system has to be investigated. To determine this effectiveness, usability measures should be specified. These measures may include the number of errors, task completion time, the user's attitude towards the interface, etc. A very important but subjective usability criteria is user satisfaction. Since the user needs to work with the interface they have to say about whether it is a good design and is pleasant to work with.

The fourth step is **Refinement and tools**.

Based on the problems encountered in the evaluation stage, a number of design improvements will be made to the current prototype. Then, a new round of design, implementation, and evaluation is started. This iterative process will run until the result is satisfactory. If proven successful the final interface technique of metaphor can be incorporated into existing user interface design tools.

Kong et al. [11] proposed an approach to develop adaptive multimodal interfaces. The approach quantifies the user preference of each modality. The framework takes the specification of interaction contexts (user, device, and environment), the modality space, the requirements of each modality, and a mapping between modality space and preference space as inputs. The approach proposes the following steps to design the interface:

- to analyze problems and design the modality space (available input/output modalities) — to elicit user requirements and identify problems;
- to determine interaction contexts — to create interaction scenarios and determine interaction contexts (user, device, and environment);
- to assess the preference score of a modality under an interaction context — to evaluate and quantify (using a formula) the preference score of a modality.

A **framework for user interface adaptation** is proposed for the development of context-sensitive user interfaces [12]. It includes six steps:

- user interface modeling (description of the abstract user interface);
- default user interface design (default version of a concrete user interface);
- supplemental user interface1 design (extend or replace the default user interface) — this step is omitted when the system generate the default user interface automatically;
- context of usage instantiation (identification and instantiation of the context of usage — user model, device model, and environment model -— by the platform);
- user interface accommodation — system-drive -— (adaption of the user interface at runtime to match a particular context of usage);
- user interface customization — user-drive — (customization of the user interface by user operations).

For each step, the authors represent a description and examples of methods and techniques that can be used through the user interface development.

Based on the analysis of existing design techniques for adaptive intelligent multimodal interfaces, we can conclude that there are no generally accepted methods and design tools, while it is possible to identify common stages that are proposed by all authors:

- analysis of the context of usage and user problems;

- interface design and development;
- evaluating the quality of the designed interface.

Disadvantages when designing user interfaces are:

- the knowledge on each stage of design is held by different specialists in an unformalized, non-uniformized form;
- the absence of a formalized documentation phase of the design steps leads in the future to the need to create separate help-systems for users, developers, etc.;
- lack of comprehensive automation of the interface design process.

III. Proposed approach

To eliminate the disadvantages of existing solutions, it is proposed to use the ontological approach based on a semantic model in the design and implementation of an adaptive intelligent multimodal user interface. Such an interface is proposed to consider as a specialized subsystem for solving user interface problems, consisting of a knowledge base and a problem solver of interface problems. It is proposed to describe the model of knowledge base and problem solver on the basis of a universal unified language of knowledge representation, which will ensure compatibility between these components.

The architecture of the interface of such a system was considered in [13]. The proposed methodology for designing adaptive intelligent multimodal user interfaces will include:

- analysis of the user, their needs and purposes, and the context of usage;
- analysis of user interface requirements;
- user interface modeling;
- default user interface design;
- development of the user interface;
- analysis of the user interface and its adaptation.

Since knowledge about a particular stage is usually held by different experts, a **feature** of the proposed approach is the **necessary** formalized documentation of knowledge in a unified form and the usage of the component approach at each of the stages.

A library of reusable components of the knowledge base, problem solver, and interface is proposed for the component approach.

Thus, **results of the first stage**, such as the model of a particular user, their needs and the context of system usage (device, environment) should be formalized within the appropriate knowledge base ontologies of the intelligent interface. In this process of formalization, if necessary, components of the knowledge base should be reused from the library of reusable components and new components can be added to the same library.

Results of the second step are the final requirements for the interface, which must be formulated with respect to the user model and its purpose, as well as with respect to the context of usage. The results should also be formalized, and existing knowledge base components from a reusable component library can be used in the execution process.

In accordance with the requirements for the user interface, a model of an adaptive intelligent multimodal user interface is constructed, which is the **result of the third stage**. Such a model will include a formalized model of the knowledge base and the problem solver.

The **result of the fourth step** is a model-based designed user interface. Interface, knowledge base, and problem solver components can be used in the design. Such components will be written in a unified form, which will ensure their automatic compatibility.

The **result of the fifth step** is the implementation of the designed user interface. In this case it is necessary to use ready interface components from the library of reusable interface components.

At the **stage of user interface analysis and adaptation**, ready-made components of the problem solver are used.

This will form a knowledge base of the designed interface, which can automatically be used as a **help-system** for users, developers, etc.

Thus, based on the above, the following demands can be made to the technology, on the ground of which this approach can be implemented:

- the technology should support a component approach to creating semantic models;
- the technology should allow the simple integration of various semantic models within a unified system;
- the technology should provide an opportunity to describe different semantic models and their components of various types of knowledge in a single format.

Among the existing system design technologies, the *OSTIS Technology* meets the specified requirements, among the advantages of which it is also possible to additionally highlight the presence of a basic set of ontologies that can serve as the ground for the IUI model being developed.

Thus, within this approach, in the article, an option for implementing a framework for building UIs is proposed, which is based on the *OSTIS Technology*, providing a universal language for the semantic representation (encoding) of information in the memory of intelligent computer systems, called an *SC-code*. Texts of the *SC-code* (sc-texts) are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-connectors* and *sc-nodes*, which, in turn, can be *sc-edges* or *sc-arcs*, depending on the orientation). The *Alphabet of the SC-code* consists of five main elements, on the ground of which SC-code constructions of any complexity are built, as well as more particular types of sc-elements are introduced (for example, new concepts). The memory that

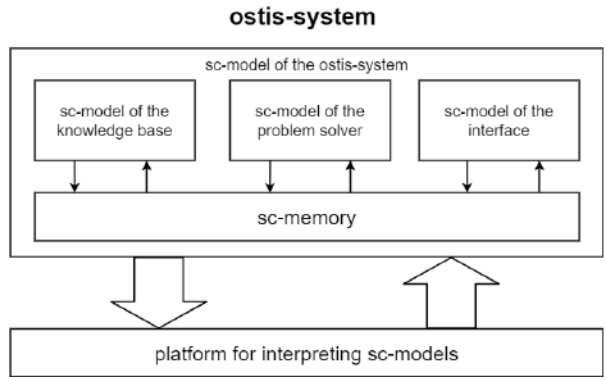

ostis-system
sc-model of the ostis-system
sc-model of the knowledge base
sc-model of the problem solver
sc-model of the interface
sc-memory
platform for interpreting sc-models

Figure 1. The architecture of the ostis-system

stores SC-code constructions is called semantic memory, or *sc-memory* [14].

The architecture of each ostis-system includes a platform for interpreting semantic models of ostis-systems as well as a semantic model of the ostis-system described using the SC-code (sc-model of the ostis-system). In turn, the sc-model of the ostis-system includes the sc-model of the KB, the sc-model of the interface, and the sc-model of the problem solver. The principles of the design and structure of KBs and problem solvers are discussed in more detail in [15] and [16], respectively. Within this article, the sc-model of the UI will be considered, which is included in the sc-model of the interface. Its principles were described in the article [17], the development of which is this work.

The architecture of the ostis-system is shown in Figure 1.

A library of reusable ostis-system components already exists within the OSTIS Technology.

It is important to note that all of its components are compatible with each other and stored in a single form of representation.

Within library of reusable ostis-system components there is the following hierarchy of components.

reusable ostis-systems component
⇒ *subdividing**:
{● *reusable knowledge base component*
⊃ *semantic neighborhood*
⊃ *subject domain and ontology*
⊃ *knowledge base*
⊃ *template of a typical ostis-systems component*
∋ *Template for the subject domain description*
∋ *Template for the relation description*
● *reusable problem solver component*
⊃ *atomic knowledge processing agent*
⊃ *knowledge processing program*
● *reusable interface component*
⊃ *reusable user interface component for display*
⊃ *interactive reusable user interface component*
}

Any ostis-system can integrate an intelligent interface according to the proposed architecture. But it is also important to clarify the concept of user interface in the context of the OSTIS Ecosystem.

Within the OSTIS Ecosystem, there is the concept of a personal ostis-assistant, an ostis-system that is a personal assistant to the appropriate human who is a part of the OSTIS Ecosystem, i.e. an ostis-system that mediates the human interactions with the members of all the collectives (ostis-communities) of which the human is a member.

Since user interaction with the OSTIS Ecosystem only takes place via a personal assistant, an adaptive intelligent multimodal user interface is required not for all ostis-systems but only for ostis-systems that are personal assistants.

A model of the user, their activities, etc. in this context should only be stored within the user's personal assistant and shared with other systems as needed.

The personal assistant must be able to retrieve the interface model of other ostis-systems and display it to the user.

Proposed approach will allow:

- unifying the methods and tools for designing user interfaces, providing the ability to reuse already developed components;
- ensuring the extensibility of the interface components;
- designing tools to help the user to interact with the interface of the system in connection with the design phase of the interface itself;
- ensuring that interface design tools and the system for which it is designed will be compatible, providing effective integration of any interface into any system;
- using the help system, which is an intermediary in communicating with the system.

IV. Conclusion

In the article, the methods of designing interfaces of next-generation intelligent computer systems are considered.

As a result for the analysis of existing methods of designing adaptive intelligent multimodal user interfaces, it was concluded that there are no generally accepted methods and means of designing user interfaces, however, there are the following general stages of design:

- analysis of the context of usage and user problems;
- interface design and development;
- evaluating the quality of the designed interface.

Among the disadvantages of the reviewed methodologies for the design of user interfaces were the following:

- the knowledge on each stage of design is held by different specialists in an unformalized, non-uniformized form;
- the absence of a formalized documentation phase of the design steps leads in the future to the need to create separate help-systems for users, developers, etc.
- lack of comprehensive automation of the interface design process.

To address these disadvantages, within the article, it is proposed to introduce a necessary stage of the formalized documentation of knowledge in a unified form, as well as an ontological approach based on a semantic model in the design and implementation of an adaptive intelligent multimodal user interface based on the OSTIS Technology is considered, which will allow:

- unifying the methods and tools for designing user interfaces, providing the ability to reuse already developed components;
- ensuring the extensibility of the interface components;
- designing tools to help the user to interact with the interface of the system in connection with the design phase of the interface itself;
- ensuring that interface design tools and the system for which it is designed will be compatible, providing effective integration of any interface into any system;
- using the help system, which is an intermediary in communicating with the system.

V. Acknowledgment

The authors would like to thank the research group of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for its help in the work and valuable comments.

References

[1] J. Foley, W. Chul, S. Kovacevic, and K. Murray, “The user interface design environment,” *ACM SIGCHI Bulletin*, vol. 20, pp. 77–78, 07 1988.

[2] T. A. Fomina and G. M. Novikova, “Proektirovanie adaptivnogo interfejsa is dlya podderzhki deyatel’nosti obrazovatel’nogo uchrezhdeniya,” *Vestnik Altajskoj akademii ekonomiki i prava*, vol. 6, no. 1, pp. 125–133, 2020.

[3] I. M. Ismagilova and S. Valeev, “Postroenie dinamicheskih adaptivnih interfeisov informacionno-upravlyayuschih sistem na osnove metodov iskusstvennogo intellekta,” *Vestnik Ufimskogo gosudarstvennogo aviacionnogo tehnicheskogo universiteta*, vol. 9, pp. 122–130, May 2018.

[4] S. Valeev, “Postroenie adaptivnih interfeisov v slojnih raspredelennih tehnicheskih sistemah s primeneniem statisticheskih metodov,” *Vestnik Ufimskogo gosudarstvennogo aviacionnogo tehnicheskogo universiteta*, vol. 9, pp. 140–150, May 2018.

[5] M. Montero and E. Gaudioso, *Adaptable and Adaptive Web-Based Educational Systems : Encyclopedia of human computer interaction*. UK: Liverpool John Moores University, 2005.

[6] E. Schlungbaum, “Individual user interfaces and model-based user interface software tools,” in *IUI ’97*, 1997.

[7] S. Brdnik, T. Heričko, and B. Šumak, “Intelligent user interfaces and their evaluation: A systematic mapping study,” *Sensors*, vol. 22, no. 15, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/15/5830

[8] P. Ehlert, *Intelligent User Interfaces: Introduction and Survey*, 02 2003.

[9] D. Benyon, “Adaptive systems: A solution to usability problems,” *User Modeling and User-Adapted Interaction*, vol. 3, pp. 65–87, 1993.

[10] H. C. Lu, “Designing the user interface: Strategies for effective human computer interaction (3rd ed.) by ben shneiderman 1998, 639 pages, $47.29 reading, ma: Addison-wesley isbn 0-201-69497-2,” *Ergonomics in Design*, vol. 6, no. 4, pp. 31–32, 1998.

[11] J. Kong, W. Zhang, N. Yu, and X. Xia, “Design of human-centric adaptive multimodal interfaces,” *Int. J. Hum.-Comput. Stud.*, vol. 69, pp. 854–869, 12 2011.

[12] G. Zimmermann, G. Vanderheiden, and C. Strobbe, “Towards deep adaptivity – a framework for the development of fully context-sensitive user interfaces,” 06 2014, pp. 299–310.

[13] M. E. Sadouski, “Ontological approach to the building of semantic models of user interfaces,” *Otkrytye semanticheskie tehnologii proektirovanija intellektual’nyh sistem [Open semantic technologies for intelligent systems]*, pp. 105–116, 2021.

[14] V. Golenkov, N. Gulyakina, I. Davydenko, and D. Shunkevich, “Semanticheskie tekhnologii proektirovaniya intellektual’nyh sistem i semanticheskie associativnye komp’yutery [Semantic technologies of intelligent systems design and semantic associative computers],” *Otkrytye semanticheskie tehnologii proektirovanija intellektual’nyh sistem [Open semantic technologies for intelligent systems]*, pp. 42–50, 2019.

[15] I. Davydenko, “Semantic models, method and tools of knowledge bases coordinated development based on reusable components,” in *Otkrytye semanticheskie tehnologii proektirovanija intellektual’nyh sistem [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed., BSUIR. Minsk , BSUIR, 2018, pp. 99–118.

[16] D. Shunkevich, “Agentno-orientirovannye reshateli zadach intellektual’nyh sistem [Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems],” in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2018, pp. 119–132.

[17] A. Boriskin, M. Sadouski, and D. Koronchik, “Ontology-based design of intelligent systems user interface,” vol. 12, pp. 95–106, 02 2017.

Методика и средства компонентного проектирования интерфейсов интеллектуальных компьютерных систем нового поколения

Садовский М.Е., Жмырко А.В.

В статье рассматривается методика проектирования интерфейсов компьютерных систем нового поколения. Описаны этапы проектирования адаптивных интеллектуальных мультимодальных пользовательских интерфейсов и применение этих этапов в контексте Технологии OSTIS.

Received 28.10.2022

Universal model of interpreting logical-semantic models of intelligent computer systems of a new generation

Daniil Shunkevich
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: shunkevich@bsuir.by

***Abstract*—In the article, an approach to solving the problem of the platform independence of computer systems is considered, which assumes unification of the principles for the implementation of such systems and ensuring their semantic compatibility based on the OSTIS Technology. The formalized system of concepts is given, that defines the principles of the implementation of this approach, including the principles for the implementation of the hardware platform for the implementation of systems built on the basis of the OSTIS Technology – an associative semantic computer.**

***Keywords*—OSTIS Technology, platform independence, ontology, associative semantic computer.**

I. Introduction

In general, the development of any artificial system, in particular, an *intelligent computer system*, involves the execution of two stages:

- the design stage, that is, the building of a formal model of the system, sufficient to understand the principles of its configuration and perform the subsequent stage of its implementation;
- the implementation stage, that is, the direct realization of the developed model using specific means (tools, materials, components, etc.). In the case of computer systems, the execution of this stage usually involves the selection of particular programming languages, libraries, third-party tools such as DBMS and various services, etc., as well as the programming and debugging of the system using the chosen means.

For each of these stages, distinct methods as well as automation tools for the corresponding processes may exist.

If the design stage of a computer system usually requires the participation of highly qualified specialists and experts in the subject domains in which automation is carried out, then the implementation stage, on the one hand, is usually simpler (in case of high-quality execution of the design stage) and, on the other hand, requires significant resources. One of the reasons for this is the need for a computer system to work on various platforms (devices), each of which, in general, may have its own features and limitations that need to be taken into account at the implementation stage. The solution to this problem is to ensure the platform independence (or cross-platform compatibility) of the computer systems being developed.

II. Analysis of modern approaches to ensuring the platform independence

The idea of ensuring the platform independence is widely used in modern computer systems for a long time. This problem is usually considered at two levels:

- the problem of enabling the work of the software system in different operation systems;
- the problem of ensuring the compatibility of the operation system with various hardware architectures. To solve this problem, different builds of the operation system kernel may exist for various hardware architectures, which is typical for Linux operation systems. At the same time, it is essential to note that in the vast majority of cases this entails not fundamentally different architectures but options for implementing the basic von Neumann architecture.

In the case when the computer system being developed is designed at a lower level than the operation system as such (for example, when programming controllers for managing various devices), the problem of ensuring the platform independence is significantly aggravated and can most often be solved only for a set of hardware of a certain class for which the access interface is standardized, that is, a set of low-level information processing commands.

Thus, it can be said that much attention in the design of modern computer systems is currently being paid to the first of the listed levels of the platform independence, that is, ensuring the operation of the software system on different operation systems. This can be achieved in different ways:

- The usage of cross-platform programming languages, which, in turn, can be divided into "fully" interpreted languages (Python, JavaScript and languages based on it, PHP, and others) and languages using compilation into the platform-independent low-level bytecode with its possible subsequent compilation into the machine code directly during execution (Just-in-time compilation, or JIT compilation). Languages of the second class include, for example, Java and C#. The implementation of this approach requires the installation of the appropriate programming language or bytecode on the target computer with the operation system of the interpreter.

Despite its popularity, this option has a number of limitations:

- on average, the performance of interpreted programs is lower than compiled ones. One of the approaches to solving this problem is JIT compilation;
- strictly speaking, cross-platform compatibility with this option is provided not for all operation systems but for a class of operation systems and the corresponding class of devices, for example, operation systems designed for personal computers. For example, an application for a personal computer written in Java cannot be directly transferred to a mobile device, because when developing mobile applications, other principles of user interaction with the system interface, the absence of multiwindowing, and much more are taken into account.

- The implementation of the system in the form of a web application, which is operated through a web browser and whose interface is thus implemented on the basis of generally accepted standards of the World Wide Web (HTML, CSS, JavaScript and languages and libraries based on it). This option provides the ability to work with the application from any device that has a web browser, including a mobile one. The disadvantages of this option include:
  - as a rule, high demands on the performance of the end device. A modern web browser is one of the most resource-intensive applications on almost any device;
  - the problem of ensuring the platform independence of the server part of the web application remains behind the scenes, which should be solved in some other way;
  - despite standardization, developers often have to take into account the specifics of particular web browsers and test the performance of applications for each of them;
  - potentially, the same web application can be used on any device, however, to ensure convenience and clarity, as a rule, it is necessary to develop separate versions of the web application adapted to different devices, having, for example, different screen sizes.
- Virtualization (containerization, emulation). The listed terms are not completely synonymous but generally denote an approach in which an isolated local environment (virtual machine, container, emulation environment) is created within the operation system, containing all the settings necessary for the work of an application and guaranteeing its work on any operation systems and devices where the corresponding virtual machine or container can be interpreted. Accordingly, the running of such environments requires the installation of an appropriate interpreter or emulator on the end device.

  This approach is rapidly developing and gaining popularity at the moment, since it allows not only solving the problem of cross-platform compatibility but also saving the consumer from installing a large number of dependencies and configuring the application on the end device.

  Among the popular tools implementing this approach, tools for virtualization (VirtualBox, DOSBox, VMware Workstation), containerization (Docker), emulation of Android applications for desktop operation systems (Genymotion, Bluestacks, Anbox), and many others can be specified.

  The disadvantages of this approach include its resource intensity and reduced performance, as well as limited usage (as a rule, the corresponding interpreters are developed only for the most popular and demanded operation systems). In addition, there is a next-level problem associated with dependence on the selected virtualization (containerization) tool.

It is also important to note that even for interpreted programming languages, there is a problem of application dependence on the set of libraries and frameworks used. So, when developing an interface of a web application, the popular AngularJS and Reactos frameworks can be used, while after selecting one of them, it is impossible to quickly transfer the application to another framework.

Thus, it can be concluded that a lot of attention is paid to the problem of ensuring the platform independence in modern computer systems, but it has not been fully solved. At the same time, there are a large number of successful private solutions, which, however, have serious limitations, primarily due to the lack of unification of modern approaches to the development of computer systems.

The problem of ensuring the platform independence becomes even more urgent in the context of the development of *intelligent computer systems*. This is conditioned by the following features of such systems:

- a much more complex structure of the represented information in comparison with traditional computer systems and, accordingly, the variety of forms of its representation, storage and processing of which on different platforms can be organized in completely different ways;

- high performance requirements for some classes of systems, in particular, systems that use machine learning, which leads to the creation of specialized hardware architectures, such as, for example, neurocomputers [1], [2];
- a variety of problem-solving models that are generally implemented differently in various systems;
- the relevance of the development of hybrid intelligent systems [3], within which various types of knowledge and various problem-solving models are integrated. Due to the lack of a generally accepted unified foundation for their integration at the moment, such systems are created mainly with a focus on a specific platform and can hardly be transferred to other platforms.

Thus, we can say that the problem of ensuring the platform independence for intelligent systems is largely conditioned by the deficiency in the semantic compatibility of components of such systems with each other, which, in turn, creates obstacles even for the implementation of approaches to ensuring the platform independence, implemented in the development of traditional computer systems.

III. Proposed approach to ensuring the platform independence of intelligent computer systems

To solve the problem of ensuring the platform independence of intelligent systems, as it was mentioned earlier, it is necessary first to ensure the semantic compatibility of the components of such systems with each other, which, in turn, assumes:

- unifying the representation of various kinds of information stored in the knowledge bases of such systems;
- unifying the basic models of processing information stored in the knowledge bases of such systems, that is, the allocation of a universal low-level programming language that allows processing the stored information in a unified form;
- unifying the principles of implementing various problem-solving models and, as a result, the possibility of their integration within hybrid intelligent systems;
- unifying the principles of developing computer system interfaces, which would make it possible to carry out within one intelligent system the interaction with other systems and users of such systems in different external languages, including natural ones.

These principles are implemented within the Open Semantic Technology of Intelligent Systems Design (*OSTIS Technology*) [4], which is proposed to be the basis for solving the problem of ensuring the semantic compatibility of components of *intelligent computer systems* and ensuring the platform independence of such systems. In particular, within the *OSTIS Technology*, the following key principles from the point of view of ensuring the platform independence are implemented:

- the *OSTIS Technology* is based on a universal method of semantic representation (encoding) of information in the memory of intelligent computer systems, called an *SC-code*. Texts of the *SC-code* (sc-texts, sc-constructions) are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, depending on orientation, can be *sc-arcs* or *sc-edges*). The *Alphabet of the SC-code* consists of five main elements, on the basis of which SC-code constructions of any complexity are built, including more specific types of sc-elements (for example, new concepts). Universality and uniformity of the *SC-code* makes it possible to describe on its basis any *types of knowledge* and any problem-solving *methods*, which, in turn, significantly simplifies their integration within a single system;
- the basis of the knowledge base developed by the *OSTIS Technology* is a hierarchical system of semantic models of *subject domains* and *ontologies*, among which the universal *Kernel of the knowledge base semantic models* and the methodology for the development of semantic knowledge base models are allocated, which ensure the semantic compatibility of the knowledge bases being developed;
- the basis of information processing within the *OSTIS Technology* is the *SCP Language*, the program texts of which are also written in the form of SC-code constructions;
- the problem solver architecture within the *OSTIS Technology* is based on a multi-agent approach, in which agents interact with each other purely by specifying the actions they perform within a common semantic memory (such agents are called *sc-agents*). Such an approach allows ensuring the fundamental possibility of implementing any *problem-solving methods* in the form of corresponding solver components and ensuring their semantic compatibility;
- the interface of the *ostis-system* is interpreted as a specialized subsystem that is built on the same principles as any other ostis-system (that is, it has its own knowledge base and problem solver) and solves problems related to the interaction of the system with the external environment;
- all of these principles together make it possible to ensure the semantic compatibility and simplify the integration of both various components of computer systems and such systems themselves.

The listed principles allow concluding that the *OSTIS*

Technology provides a fundamental possibility of implementing the platform independence of computer systems developed on its basis (*ostis-systems*). On the other hand, thanks to its universality, the *OSTIS Technology* allows transforming any modern computer system into the *ostis-system*, which will be functionally equivalent to the original computer system but at the same time will have all the above features that create preconditions for solving the problem of the platform independence.

To solve this problem at the level of the *OSTIS Technology* it is proposed to use an ontological approach involving the building of a family of *ontologies*, providing clarification of concepts such as *ostis-system*, *ostis-platform*, their structure, typology, and the requirements imposed on them.

As for the above-mentioned problem of the dependence of computer systems on specific frameworks, a similar problem may arise with the further development of the *OSTIS Technology*, in a situation where the corresponding libraries will contain a sufficiently large number of functionally equivalent components. However, thanks to the principles underlying the *OSTIS Technology*, in particular, the semantic representation of information and semantic compatibility of components, this problem will be much less acute, since:

- the number of functionally equivalent components will be significantly lower than in traditional information technologies; it is not necessary to create syntactically different components: the differences will be only at the semantic level;
- independently, the components will be more universal, that is, they can be used in a much larger number of systems;
- there is an opportunity to automatically identify close components, their similarities, differences, potential conflicts, and dependencies of components;
- it is possible to build fairly simple (compared to traditional technologies) procedures for the transition from one framework to another, since all components and frameworks have a common formal semantic basis of a level that is higher than in traditional technologies.

Within the *OSTIS Technology*, several universal variants of visualization of *SC-code* constructions are proposed, such as *SCg-code* (graphic variant), *SCn-code* (nonlinear hypertext variant), *SCs-code* (linear string variant). Within this article, fragments of structured texts in the SCn code [4] will often be used, which are simultaneously fragments of the source texts of the knowledge base, understandable to both human and machine. This allows making the text more structured and formalized, while maintaining its readability. The symbol ":=" in such texts indicates alternative (synonymous) names of the described entity, revealing in more detail certain of its features.

IV. Architecture and principles for the ostis-systems implementation

Let us consider the proposed approach to organizing the implementation of *ostis-systems*. One of the key principles of the *OSTIS Technology* is to ensure the platform independence of *ostis-systems*, that is, a strict separation of the logical-semantic model of the cybernetic system (*sc-models of the cybernetic system*) and the interpretation platform of the sc-models of the cybernetic system (*ostis-platform*). The advantages of such a strict separation are quite obvious:

- the transfer of the *ostis-system* from one platform to another (for example, a newer and more efficient or focused on a certain class of devices) is performed with minimum overhead costs (in the ideal case, it generally comes down to loading the *sc-model of a cybernetic system* onto the platform);
- the components of *ostis-systems* become universal, that is, they can be used in any ostis-systems where their usage is appropriate;
- the development of the platform and the development of sc-models of systems can be carried out in parallel and independently of each other, in general, by separate independent teams of developers according to their own rules and methods.

Let us consider in more detail the concept of the *logical-semantic model of a cybernetic system*.

logical-semantic model of a cybernetic system

:= [formal model (formal description) of the functioning of a cybernetic system, consisting of (1) a formal model of information stored in the memory of a cybernetic system and (2) a formal model of a group of agents processing the specified information]

$\supset$ *sc-model of a cybernetic system*

    := [logical-semantic model of a cybernetic system, represented in an SC-code]

    := [logical-semantic model of an ostis-system, which, in particular, can be a functionally equivalent model of some cybernetic system that is not an ostis-system]

cybernetic system

$\supset$ *computer system*

    := [artificial cybernetic system]

    $\supset$ *ostis-system*

        := [computer system built using the OSTIS Technology based on the interpretation of the designed logical-semantic sc-model of this system]

ostis-system
⊂ *subject*
⇒ *generalized decomposition**:
{• *sc-model of a cybernetic system*
• *ostis-platform*
}

sc-model of a cybernetic system
⇒ *generalized decomposition**:
{• *sc-memory*
• *sc-model of the knowledge base*
• *sc-model of the problem solver*
• *sc-model of the cybernetic system interface*
}

sc-memory
:= [abstract sc-memory]
:= [sc-storage]
:= [semantic memory storing SC-code constructions]
:= [storage of SC-code constructions]

The ***sc-memory*** is, on the one hand, a common environment for storing the *knowledge base* and, on the other hand, an environment for interaction of *sc-agents*. At the same time, each *sc-agent* relies on some known *sc-elements* stored in the *sc-memory* (*key sc-elements* of this *sc-agent*).

In general, the *sc-memory* implements the following functions:

- storage of *SC-code* constructions;
- storage of information constructions (files) external to the *SC-code*. In general, file storage can be implemented in a way different from storing sc-constructions;
- access to *SC-code* constructions (reading, creating, deleting), implemented through the appropriate software or hardware interface. Such an interface is essentially a microprogramming language that allows implementing more complex procedures for processing stored constructions based on it, including the operators of the *SCP Language*, the set of which determines the list of commands of such a microprogramming language. The *sc-memory* itself is passive in this regard and only executes commands initiated from the outside by any subjects.

Note that the separation of the storage and access functions is rather conditional, since it seems impractical to implement the function of storing constructions without the possibility of accessing them at least at the lowest level, because it will be impossible to use such storage.

The terms "*sc-memory*" and "*abstract sc-memory*" are synonyms in the way that mentioning the *sc-memory* we mean some abstraction for which its maximum volume is not specified (the maximum number of *sc-elements*, that can be stored in this memory simultaneously), a particular method of storing *sc-elements*, means for ensuring the storage reliability, etc. All these features are specified at the level of the *sc-memory implementation* in a hardware version or a software model based on some other architecture.

The explicit allocation of the *sc-model of the knowledge base*, *sc-model of the problem solver*, and *sc-model of the cybernetic system interface* within the *sc-model of the cybernetic system* is to a certain extent conditional, since to ensure the platform independence, *sc-models of a cybernetic system*, the *problem solver*, and the *system interface* are described by means of the *SC-code* and, thus, are also part of the *knowledge base*. Such an explicit allocation of these components is conditioned by the convenience of designing and maintaining the system.

Thus, on condition of strict separation of the *sc-model of the cybernetic system* and *ostis-platform*, as well as ensuring the universality of the *ostis-platform*, that is, the ability to interpret any *sc-model of the cybernetic system* on any variant of the *ostis-platform*, the implementation stage of the *ostis-system* actually comes down to loading the *sc-model of a cybernetic system* on the selected variant of the *ostis-platform*.

It is important to note that the universality of a particular implementation option of the *ostis-platform* is obviously limited by the physical (hardware) part of this implementation. For example, if the hardware of the selected platform option is a conventional personal computer, then without the addition of extra hardware components, the system will not be able to solve problems related to the physical movement of itself and other objects in space, even if the software part of the system is able to perform the necessary calculations. In other words, any *ostis-platform* will always be limited in solving *behavioral problems* of any classes, no matter how powerful physical resources it possesses. Thus, it is more correct to talk about the universality of the *ostis-platform* in the context of solving *information problems*, that is, the ability to interpret any *sc-models of cybernetic systems* regardless of what kind of *information problems* these systems solve.

Based on this, it is possible to formulate a key requirement for the *sc-model of a cybernetic system* – at none of the stages of solving any *information problem* in this system the features of the platform on which the specified *sc-model* will be interpreted in the future should be taken into account. Similarly, the key requirement for the *ostis-platform* is to provide an interface for accessing (searching and converting) information stored in the *sc-memory* in some universal way, independent from the specifics of the implementation of a particular platform. Thus, the most important problem to ensure the platform independence of *ostis-systems* is a clear specification of the requirements for each implementation of the *ostis-platform*, that is, standardization of *ostis-platforms*. It is important to note that such standardization should not depend on the form in which the *ostis-platform* is implemented and, accordingly, be suitable for the hardware version of the implementation.

To clarify the requirements for the *ostis-platform*, we introduce the concept of an *sc-machine*, which is an analogue of such models as the Post Machine and the Turing Machine [5], the von Neumann Machine [6].

sc-machine

:= [abstract sc-machine]

:= [generalization of various implementations of ostis-platforms, for which general functional requirements are set]

:= [generalized model describing the functioning of any ostis-platform, regardless of the way it is implemented]

:= [generalized model that defines the general patterns of any ostis-platform, regardless of the way it is implemented]

:= [generalized information image of the ostis-platform]

⇐ *generalized model**:
ostis-platform

⇒ *generalized decomposition**:

{• *sc-memory*

⇐ *generalized model**:
implementation of the sc-memory

• *abstract machine of knowledge processing*

⊂ *abstract sc-agent*

}

⊃ *scp-machine*

⇐ *generalized model**:
scp-interpreter

:= [sc-machine that provides interpretation of the ostis-systems basic programming language]

:= [generalized model of the interpreter of the ostis-systems basic programming language]

:= [generalized model defining the general principles for the interpretation of the ostis-systems basic programming language]

:= [generalized model of operational semantics of the ostis-systems basic programming language]

Potentially, we can talk about several possible functionally equivalent variants of the *scp-machine*, which will correspond to different variants of the basic programming language. Within the current version of the *OSTIS Technology*, both the denotational semantics of the *SCP Language* and its operational semantics, implemented in the form of an *abstract scp-machine*, are fixed [4].

It is important to emphasize that despite the advantages of the platform-independent implementation of *ostis-systems*, it sometimes turns out to be advisable to implement some components of *ostis-systems* (for example, specific *sc-agents* or user interface components) at the level of the *ostis-platform*. In the case of such an implementation of the *sc-agents* programs, an analogy can be drawn with the implementation of any subprograms at the level of microprogramming languages for modern computers. Most often, the reasonableness of such a solution is conditioned by an increase in the performance of such components and the system as a whole, since the implementation of the component, taking into account the features of the platform, is generally more productive. At the same time, let us note that the latter statement is not always true, since when implementing a component at the level of a logical-semantic model, for example, parallel information processing models can be implemented, which are not always easily and clearly implemented at the platform level.

Thus, when designing each specific *ostis-system*, the developer needs to make a decision about the implementation of certain components at a platform or platform-independent level. At the same time, it is obvious that from the point of view of technology development and the accumulation of project experience, the implementation of *ostis-systems* components at a platform-independent level is a higher priority.

Based on the above, we can assume the existence of *ostis-systems* in which all *sc-agents* are implemented at the platform level, which in this case is essentially "cut out" for a specific *ostis-system* and can be considered as an analogue of a specialized computer focused on solving problems of only a certain limited class. Let us call such an option for the implementation of *ostis-systems* the *minimum ostis-system configuration*. In order for *minimum ostis-system configuration* to be considered an *ostis-system* at all, that is, a system that is built in accordance with the principles of the *OSTIS Technology*, it must meet the following minimum set of requirements:

- the usage of the *SC-code* as a basic language for encoding information in the knowledge base and, accordingly, the presence of memory storing *SC-code* constructions;
- the presence of the *knowledge base* defining the denotational semantics of the concepts used by the system;
- the presence of at least one *internal sc-agent* performing knowledge processing in the memory of the *ostis-system*. This *sc-agent* can be implemented at the platform level, accordingly, the *knowledge base* of such a system may not contain procedural knowledge (methods).

Such a variant of *minimum ostis-system configuration* has only an *internal sc-agent* and, accordingly, has no ability to communicate with the external world (we can say that such an *ostis-system* does not have "sense organs"). In order for the system to be able to communicate with the external world, it is necessary to add at least one *receptor sc-agent* and at least one *effector sc-agent* to the *minimum ostis-system configuration*.

It is important to note that, as can be seen from the description of the *minimum ostis-system configuration*, in general, the *ostis-system* does not have to be an *intelligent system* by default. Usage of the *OSTIS Technology* for the development of computer systems does not automatically make them intelligent – it allows ensuring the possibility of subsequent unlimited intellectualization of such systems with minimum overhead costs, provided that all the principles of the *OSTIS Technology* are satisfied during their development.

V. Clarification of the ostis-platform concept

ostis-platform
:= [platform for interpreting sc-models of computer systems]
:= [interpreter of sc-models of cybernetic systems]
:= [interpreter of unified logical-semantic models of computer systems]
:= [family of platforms for interpreting sc-models of computer systems]
:= [platform for implementing sc-models of computer systems]
:= [embedded empty ostis-system]
:= [sc-machine implementation]
⊂ *embedded ostis-system*
⊂ *platform-dependent reusable component of ostis-systems*

The implementation of the *ostis-platform* (interpreter of sc-models of cybernetic systems) can have a large number of variants – both software and hardware implemented. If necessary, any components of problem solvers or knowledge bases can be included in the ***ostis-platform*** in advance at the platform-dependent level, for example, in order to simplify the creation of the first version of an *applied ostis-system*. The implementation of the *ostis-platform* can be carried out on the basis of an undefined set of existing technologies, including the hardware implementation of any of its parts. From the point of view of the component approach, any ***ostis-platform*** is a ***platform-dependent reusable component of ostis-systems***.

ostis-platform
⇒ *subdividing**:
{• *basic ostis-platform*
:= [basic interpreter of logical-semantic models of ostis-systems]
:= [minimum universal ostis-platform that provides interpretation of the sc-model of any *ostis-system* and includes an interpreter of the *ostis-systems* basic programming language (SCP Language)]
:= [universal interpreter of sc-models of ostis-systems]
:= [universal basic ostis-system that provides an imitation of any *ostis-system* by interpreting the sc-model of the imitated ostis-system]
• *extended ostis-platform*
:= [ostis-platform containing additional components implemented at the platform level]
:= [basic ostis-platform and many components implemented at the platform level]
• *specialized ostis-platform*
:= [ostis-platform that does not contain an implementation of the SCP language interpreter]
:= [non-universal ostis-platform]
}

The concept of a *basic ostis-platform* is key from the point of view of ensuring the platform independence of *ostis-systems*. The universality of the *basic ostis-platform* implies the possibility of interpreting any *sc-model of a cybernetic system* based on it. This is accomplished by the presence of means within the *OSTIS Technology*, that allow describing the *knowledge base*, *problem solver*, and *cybernetic system interface* at the level of the sc-model, as well as by the availability of a Basic universal programming language for *ostis-systems* (*SCP Language*). In this case, the *SCP Language* acts as a basic low-level standard (assembler) for processing *SC-code* constructions, guaranteeing completeness from the point of view of processing, that is, providing the ability to perform any transformation of any fragment of the *SC-code* on condition that the syntactic correctness of this fragment is maintained. It should be noted that in general there may be several such functionally equivalent assemblers (and, as a consequence, corresponding *scp-machines*), but to ensure compatibility within the *OSTIS Technology* one of these options is selected as a standard and described in the corresponding section of the *OSTIS Standard* [4].

Thus, the main and only requirement imposed on all *basic ostis-platforms* to ensure their universality is the need to provide interpretation of the *SCP Language* standardized within the *OSTIS Technology*. It is important to note that all *basic ostis-platforms* must be functionally equivalent, since they interpret the same standard of the *SCP Language*.

Each *basic ostis-platform* contains:

- implementation of the means for storing *SC-code* constructions (sc-memory), including the implementation of file memory;
- implementation of tools for processing *SC-code* constructions – an *scp-interpreter*;
- implementation of a basic set of *receptor sc-agents* and *effector sc-agents*, providing the minimum necessary information exchange between the *ostis-system* and the external environment. The specific list of such agents requires clarification, however, we can say that in general they can be implemented as part of the *scp-interpreter* (in this case, they will correspond to certain classes of *scp-operators*) or separately from it as part of the platform;
- implementation of a set of sc-agents that provide the basic functions of the *ostis-system*, related to ensuring its operation, which in principle cannot be implemented at a platform-independent level. Such functions include, for example, starting the system, loading the knowledge base into the system memory, starting the *scp-interpreter*, etc.

More formally, the model of the *basic ostis-platform* can be written as follows:

basic ostis-platform
⇒ *generalized decomposition**:
{• *sc-memory implementation*
⇒ *generalized part**:
implementation of the sc-machine file memory
• *scp-interpreter*
• *basic subsystem for interaction of the ostis-system with the external environment*
• *subsystem for ensuring the operation of the ostis-system*
}

An *extended ostis-platform* is a *basic ostis-platform* supplemented with any set of components (at least one) implemented at the platform level, provided that all the features of the *basic ostis-platform* are maintained. Thus, an *extended ostis-platform* is essentially a *basic ostis-platform* adapted to more efficiently solve problems of certain classes within a specific class of *ostis-systems*. A component implemented at the platform level becomes part of this platform and thus transforms the *basic ostis-platform* into the *extended ostis-platform*.

Introduction of the concept of the *extended ostis-platform* allows formulating a number of additional principles for the implementation of *ostis-systems*:

- there may be an undefined number of ostis-systems, each of which will have its own unique *extended ostis-platform*, but they will all be based on the same variant of the *basic ostis-platform*;
- for each variant of the *basic ostis-platform*, there may be its own *library of reusable ostis-platform components* compatible with this variant of the *basic ostis-platform* and that allows composing various variants of the *extended ostis-platform* based on the *basic ostis-platform*.

A ***specialized ostis-platform*** is a bounded implementation of the *ostis-platform* that does not contain an *scp-interpreter*. Thus, all *sc-agents*, within the *ostis-system* based on the *specialized ostis-platform*, must be implemented at the platform-dependent level. Such a *specialized ostis-platform* is an analogue of a specialized computer implemented for a specific computer system. Thus, in general, each *ostis-system* implemented on the *specialized ostis-platform* will have its unique *specialized ostis-platform*.

The ***specialized ostis-platform*** can be obtained from the *basic ostis-platform* by excluding the implementation of the *scp-interpreter* from it and implementing all necessary *sc-agents* at the platform level (or borrowing all or part of the agents from a *library of reusable ostis-platform components*, that corresponds to the given variant of the *basic ostis-platform*).

specialized ostis-platform
⇒ *generalized decomposition**:
{• *sc-memory implementation*
⇒ *generalized part**:
implementation of the sc-machine file memory
• *basic subsystem for interaction of the ostis-system with the external environment*
• *subsystem for ensuring the operation of the ostis-system*
• *specialized platform-dependent knowledge processing machine*
:= [sc-agent, as a rule, a non-atomic one, providing the performance of all the functions of some specialized ostis-platform related to knowledge processing]
⊂ *platform-dependent sc-agent*
}

The concept of the *minimum ostis-system configuration* introduced earlier can be clarified taking into account the concept of the *specialized ostis-platform*.

minimum ostis-system configuration
⇒ *generalized decomposition**:
{• *sc-model of the knowledge base*
• *specialized ostis-platform*
}

The usage of *specialized ostis-platforms* may be reasonable at the initial stage of the development of the *OSTIS Technology*, as well as in order to improve the performance of particular *ostis-systems* that are most highly loaded, however, the active development of such *specialized ostis-platforms* and their components from the point of view of the *OSTIS Technology* is impractical, since:

- if any component is designed with a focus on a specific platform, then there are no guarantees that it can be reused in other *ostis-platform* implementations options (at least, components developed for the *ostis-platform software implementation* will not be able to be used within the *associative semantic computer*);
- the availability of a large number of platform-dependent components requires the development and maintenance of a separate library infrastructure for storing and reusing such components. The greater the number of platform-dependent components and the more variants of *ostis-platforms* exist, the more complex and lengthy such an infrastructure will be. At a minimum, it will be necessary to monitor the compatibility of components with different versions of various *ostis-platforms* implementation options;
- changes in the *specialized ostis-platform*, for example, related to the transition to a newer and more efficient version of the *basic ostis-platform*, on the basis of which this *specialized ostis-platform* is built, in general, may lead to the need in making changes to components that depend on this *ostis-platform* implementation option. The more such platform-dependent components exist, the more potential changes may be required and, accordingly, the more difficult the evolution of the platform will be, provided that the operability of the *ostis-systems* in which it is used is preserved.

The above theses are also true for *extended ostis-platforms*, however, in the case of *extended ostis-platform*, problems associated with the transition to a newer version of the platform and changes in the corresponding components can always be solved by temporarily replacing platform-dependent components with their platform-independent versions with a corresponding decreased performance but maintaining the functional integrity of the system.

ostis-platform
⇒ *subdividing**:
{● *single-user ostis-platform*
:= [option for implementing a platform for interpreting sc-models of computer systems, designed for the case when only one user (owner) interacts with a particular *ostis-system*]
● *multi-user ostis-platform*
:= [option for implementing the platform for interpreting sc-models of computer systems, designed for the case when different users can interact with a particular *ostis-system* at the same time or at different times, generally having different rights, areas of responsibility, level of experience and having their own confidential part of the information stored in the knowledge base]
}

With a single-user platform implementation, it turns out to be impossible to implement some important principles of the *OSTIS Technology*, for example, the collective coordinated development of the knowledge base of the system during its operation. At the same time, various third-party tools can be used, for example, for developing a knowledge base at the level of source texts.

ostis-platform
⇒ *subdividing**:
{● *software version of the ostis-platform*
:= [platform for interpreting sc-models of ostis-systems, implemented as a software system based on traditional computer architecture]
:= [software platform for interpreting sc-models of ostis-systems]
:= [software interpreter of sc-models of ostis-systems]
● *associative semantic computer*
:= [hardware platform for interpreting sc-models of ostis-systems]
:= [hardware implemented basic interpreter of sc-models of ostis-systems]
}

It is important to note that in any *ostis-platform* implementation option, both software and hardware are always present. So, any *software version of the ostis-platform* assumes its subsequent interpretation on some hardware basis, for example, on a personal computer with a traditional architecture. At the same time, the development of the *ostis-platform* in the form of an *associative semantic computer* involves the development of a set of micro-programs implementing basic operations of searching and converting sc-constructions stored in the *sc-memory*.

Thus, the separation of the set of possible *ostis-platform* implementations into software and hardware variants rather reflects the variant of the hardware architecture on which one or another variant of the platform implementation is ultimately oriented – either the traditional von Neumann architecture or the specialized architecture of the *associative semantic computer* with structurally reconfigurable (graphodynamic) memory. In fact, the *software version of the ostis-platform* is a model (virtual machine) of the *associative semantic computer*, built on the basis of the traditional von Neumann architecture, and the *SCP Language* acts as an assembler for the *associative semantic computer* and can also be interpreted both within the hardware implementation of such a computer and within its software model.

The appropriateness of developing *ostis-platform software options* at the moment is conditioned by the obvious prevalence of the von Neumann architecture and, accordingly, the need to implement *ostis-systems* on modern computers of various types. At the same time, it is obvious that the development of specialized *associative semantic computers* will significantly increase the efficiency of *ostis-systems*, and a clear separation of the *sc-model of a cybernetic system* and its interpretation platform will allow the translation of already working *ostis-systems* from traditional architectures on *associative semantic computers* with minimum overhead costs.

Each specific *ostis-system* uniquely corresponds to a particular *ostis-platform*, which can relate to a different set of classes of *ostis-platforms*. At the same time, it is obvious that at the stage of platform development, some variant of the *ostis-platform* is designed and implemented, which is then replicated into different *ostis-systems*. Subsequently, changes can be made to this variant of the *ostis-platform* in each *ostis-system*, but in general, in a large number of *ostis-systems*, fully equivalent *ostis-platforms* can be used. Thus, it is advisable to talk about *typical ostis-platforms*, which:

- are the object of development for developers of *ostis-platforms*;
- *are a reusable component of ostis-systems* and are specified within the appropriate libraries;
- are a sample for replication (copying) when creating new *ostis-systems*.

VI. Associative semantic computers for ostis-systems

The usage of modern hardware and software platforms focused on address access to data stored in memory for the development of *ostis-systems* is not always efficient, since when developing intelligent systems, it is actually necessary to model nonlinear memory based on the linear one. Improving the efficiency of problem solving by intelligent systems requires the development of specialized platforms, including hardware ones, focused on unified semantic models of information representation and processing. Thus, the main purpose of creating *associative semantic computers* is to increase the performance of ostis-systems.

Let us consider in more detail the features for the logical organization of the traditional architecture of computer systems, which significantly complicate the effective implementation of *ostis-systems* based on it:

- low level of memory access, i.e. complexity and lengthiness of performing the procedure of associative search for the necessary knowledge fragment;
- linear memory organization and an extremely simple view of constructive objects directly stored in memory. This leads to the fact that in intelligent systems built on the basis of modern computers, the manipulation of knowledge is carried out with great difficulty. Firstly, it is necessary to operate not with the structures themselves but with their lengthy linear representations (lists, adjacency matrices, incidence matrices); secondly, the linearization of complex structures destroys the locality of their transformations;
- the information representation in the memory of modern computers has a level that is very far from the semantic one, which makes the processing of knowledge rather lengthy, requiring consideration of a large number of details concerning not the meaning of the processed information but the way it is represented in memory;
- in modern computers, there is a low level of hardware-implemented operations on non-numeric data and there is no hardware support for logical operations on knowledge fragments with a complex structure, which makes manipulating such fragments complicated.

The listed features, in fact, are not eliminated either in the approaches to build non-traditional high-performance computers (for example, computers designed for training and interpretation of artificial neural networks [1], [2]) currently being developed, because, basically, all these approaches (even if they deviate far enough from the basic principles of the organization of computing machines, proposed by von Neumann) implicitly preserve the point of view of the computer as a large arithmometer and thereby preserve its orientation to numerical tasks.

There are a number of articles [7]–[14] and patents [15]–[17] aimed at developing hardware architectures designed to process information represented in more complex forms than in traditional architectures, but they have not gained widespread distribution and application, due, firstly, to the particular solutions offered and, secondly, due to the lack of a common universal and unified coding language for any information, in the role of which, within the OSTIS Technology, the SC-code acts.

The *SC-code*, which is the formal basis of the *OSTIS Technology* was originally developed as a language for encoding information in memory of *associative semantic computers*, so it originally laid down such principles as universality (the ability to represent knowledge of any kind) and unification (uniformity) of representation, as well as minimization of the *Alphabet of the SC-code*, which, in turn, makes it easier to create a hardware platform that allows storing and processing texts of the *SC-code*.

associative semantic computer

:= [hardware implemented interpreter of semantic models (sc-models) of computer systems]

:= [semantic associative computer controlled by knowledge]

:= [computer with a nonlinear structurally configurable (graphodynamic) associative memory, the processing of information in which is reduced not to a change in the state of memory elements but to a change in the configuration of the connections between them]

:= [sc-computer]

:= [scp-computer]

:= [new generation universal computer specially designed for the implementation of semantically compatible hybrid intelligent computer systems]

:= [new generation universal computer focused on hardware interpretation of logical-semantic models of intelligent computer systems]

:= [new generation universal computer focused on hardware interpretation of ostis-systems]

:= [ostis-computer]

:= [computer for the implementation of ostis-systems]

:= [computer controlled by the knowledge represented in the SC-code]

:= [computer focused on SC-code text processing]

Let us consider the principles underlying the implementation of *associative semantic computers*:

- nonlinear memory – each elementary fragment of a text stored in memory can be incident to an unlimited number of other elementary fragments of this text. Thus, memory cells, unlike ordinary memory, are connected not by fixed conditional connections that specify a fixed sequence (order) of cells in memory but by actually (physically) conducted connections of undefined configuration. These connections correspond to arcs, edges, hyperedges of the graph (sc-text) recorded in memory;
- structurally tunable (reconfigurable) memory – the procedure of processing information stored in memory is reduced not only to changing the state of elements but also to reconfiguring the connections between them. That is, during the processing of information in structurally-tunable memory, the changes concern not only and not even so much the states of memory cells, as in ordinary memory, as the configuration of the connections between these cells. I.e., in structurally-tunable memory, during the processing of information, not only the labels on the vertices of the graph recorded in memory are redistributed, but the structure of this graph itself is also changing;
- as an internal way of encoding knowledge stored in the memory of the *associative semantic computer*, a universal (!) method of nonlinear (graph-like) semantic representation of knowledge – SC-code – is used;
- information processing is carried out by a group of agents working on shared memory. Each of them reacts to a corresponding situation or event in memory (a computer controlled by stored knowledge);
- there are software-implemented agents whose behavior is described by agent-oriented programs stored in memory, which are interpreted by the corresponding groups of agents;
- there are basic agents that cannot be software implemented (in particular, these are agents of interpretation of agent programs, basic receptor agents-sensors, basic effector agents);
- all agents work on shared memory at the same time. Moreover, if several conditions of its usage arise for an agent at some point in time in different parts of memory, different information processes corresponding to the specified agent in different parts of memory can be performed simultaneously;
- in order for the agents' information processes running in parallel in shared memory not to "interfere" with each other, the current state is recorded and constantly updated in memory for each information process. That is, each information process informs others about its intentions and wishes, which other information processes should not interfere with. The implementation of this approach can be carried out, for example, on the basis of the mechanism of locking elements of semantic memory [4];
- the processor and memory of the *associative semantic computer* are deeply integrated and form a single processor-memory. The processor of the *associative semantic computer* is evenly "distributed" over its memory so that the processor elements are simultaneously computer memory elements. That is, each cell is supplemented by a functional (processor) element, and the tunable connections between the cells become switched communication channels between the functional elements. At the same time, each functional element has its own special internal register memory, reflecting aspects of the current state of performing elementary operations of the internal language, that are important for this

functional element.

Information processing in the *associative semantic computer* is reduced to reconfiguration of communication channels between processor elements, therefore the memory of such a computer is nothing but a switchboard (!) of these communication channels. Thus, the current state of the configuration of these communication channels is the current state of the information being processed. This principle provides a significant acceleration of information processing by eliminating the stages of transferring information from memory to the processor and back, but it is paid for at the cost of a large redundancy of functional (processor) means evenly distributed over memory.

Acknowledgment

The author would like to thank the research groups of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics and the Brest State Technical University for their help in the work and valuable comments.

The work was carried out with the partial financial support of the BRFFR (BRFFR-RFFR No. F21RM-139).

References

[1] L. G. Komarcova and A. V. Maksimov, *Neurocomputers: A Textbook for Universities. - 2nd ed., revised and enlarged*, ser. Informatics at the Technical University. Moscow: Bauman Moscow State Technical University, 2004, (In Russ).

[2] (2021, Jun) USB Accelerator | Coral. [Online]. Available: https://coral.ai/products/accelerator/

[3] A. Kolesnikov, *Gibridnye intellektual'nye sistemy: Teoriya i tekhnologiya razrabotki [Hybrid intelligent systems: theory and technology of development]*, A. M. Yashin, Ed. SPb.: SPbGTU, 2001.

[4] V. Golenkov, N. Guliakina, and D. Shunkevich, *Otkrytaja tehnologija ontologicheskogo proektirovanija, proizvodstva i jekspluatacii semanticheski sovmestimyh gibridnyh intellektual'nyh komp'juternyh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems]*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[5] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory, languages, and computation*, 2nd ed. Upper Saddle River, NJ: Pearson, Nov. 2000.

[6] M. Godfrey and D. Hendry, "The computer as von Neumann planned it," *IEEE Annals of the History of Computing*, vol. 15, no. 1, pp. 11–21, 1993. [Online]. Available: https://doi.org/10.1109/85.194088

[7] H.-N. Tran and E. Cambria, "A survey of graph processing on graphics processing units," *The Journal of Supercomputing*, vol. 74, no. 5, pp. 2086–2115, Jan. 2018. [Online]. Available: https://doi.org/10.1007/s11227-017-2225-1

[8] X. Shi, Z. Zheng, Y. Zhou, H. Jin, L. He, B. Liu, and Q.-S. Hua, "Graph processing on GPUs," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1–35, Nov. 2018. [Online]. Available: https://doi.org/10.1145/3128571

[9] Y. Lü, H. Guo, L. Huang, Q. Yu, L. Shen, N. Xiao, and Z. Wang, "GraphPEG," *ACM Transactions on Architecture and Code Optimization*, vol. 18, no. 3, pp. 1–24, Sep. 2021. [Online]. Available: https://doi.org/10.1145/3450440

[10] I. V. Afanasyev, V. V. Voevodin, K. Komatsu, and H. Kobayashi, "VGL: a high-performance graph processing framework for the NEC SX-aurora TSUBASA vector architecture," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 8694–8715, Jan. 2021. [Online]. Available: https://doi.org/10.1007/s11227-020-03564-9

[11] J. Zhang, S. Khoram, and J. J. Li, "Boosting the Performance of FPGA-based Graph Processor using Hybrid Memory Cube: A Case for Breadth First Search," *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017.

[12] Y. Hu, Y. Du, E. Ustun, and Z. Zhang, "GraphLily: Accelerating Graph Linear Algebra on HBM-Equipped FPGAs," *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2021.

[13] L. Minati, V. Movsisyan, M. Mccormick, K. Gyozalyan, T. Papazyan, H. Makaryan, S. Aldrigo, T. Harutyunyan, H. T. Ghaltaghchyan, C. Mccormick, and M. L. Fandrich, "iFLEX: A Fully Open-Source, High-Density Field-Programmable Gate Array (FPGA)-Based Hardware Co-Processor for Vector Similarity Searching," *IEEE Access*, vol. 7, pp. 112 269–112 283, 2019.

[14] W. S. Song, V. Gleyzer, A. Lomakin, and J. Kepner, "Novel graph processor architecture, prototype system, and results," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Sep. 2016. [Online]. Available: https://doi.org/10.1109/hpec.2016.7761635

[15] S. Somsubhra, "Reconfigurable semantic processor," Oct 2006.

[16] J. D. Allen, J. Philip, and L. Butler, "Parallel machine architecture for production rule systems," Jun 1989.

[17] M. Moussa, A. Savich, and S. Areibi, "Architecture, system and method for artificial neural network implementation," Jun 2013.

Универсальная модель интерпретации логико-семантических моделей интеллектуальных компьютерных систем нового поколения

Шункевич Д.В.

В работе рассматривается подход к решению проблемы платформенной независимости компьютерных систем, предполагающий унификацию принципов реализации таких систем и обеспечения их семантической совместимости на основе Технологии OSTIS. Приводится формализованная система понятий, определяющая принципы реализации данного подхода, включая прицнипы реализации аппаратной платформы для реализации систем, построенных на основе Технологии OSTIS, – ассоциативного семантического компьютера.

Received 30.10.2022

Software platform for next-generation intelligent computer systems

Nikita Zotov
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: nikita.zotov.belarus@gmail.com

***Abstract*—This paper describes the methodology of designing semantically compatible computer systems and ensuring their independence from the implementation of platforms for designing such systems. The article demonstrates the significance of designing and implementing next-generation platforms, and also proposes the solution to the problem in the form of designing and developing universal interpreters of logical-semantic models of systems according to the principles of the OSTIS Technology. This article is also a formal specification of the first software implementation of the ostis-platform.**

***Keywords*—computer aided design (CAD), ontological design, automation tools for the design and development of computer systems, graph database, graph knowledge base, database management system (DBMS), knowledge base management system, universal interpreter, graph storage, ostis-platform**

I. Introduction

The main result of research in the field of design and development of computer systems (c.s.) is not so much the existing c.s., but the development of technologies and tools based on the principles of these technologies that allow to quickly and in large quantities generate a wide variety of c.s. [1] that have great practical value. Right now, there are a wide variety of solutions in the field of automated design and development of c.s., which allow solving problems of a fairly serious level of complexity [2]. However, none of these systems are able to provide platform independence, and hence the semantic compatibility and interoperability of the created computer systems. The urgency of the problem is explained by the need to create next-generation computer systems capable of solving problems of any kind quickly and adequately [3]. To achieve this, it is necessary to design the *Standard of design and developing of c.s.* and implement such tools according to this standard [4], [5], [6].

II. Problems of the current state of computer systems and platforms for their implementation

Modern computer systems, as well as automation tools for the design and development of such systems, have a number of significant disadvantages:

1) Computer systems to a great extent remain dependent on the implementation of specific platforms on which they are designed, which, in turn, leads to significant costs for integrating the methods and tools for system design in case of transition to new platforms.
2) The design and development of the implementation of a particular system is carried out using different methods and models for designing software c.s. Thus, the description of the target state of the system and the description of the current implementation may not correspond to each other, and the integration of such solutions is difficult to achieve. This problem is well described when designing a platform for practical applications [7].
3) Specification of software systems is relegated to the background, and sometimes it is not done at all by the development project of a specific c.s. Consequently, the costs of maintaining the process of permanent re-engineering of such systems increase [8].
4) When developing modern systems, there is no understanding of the need to develop and describe the design methods for these systems, including descriptions of the implementation process, directions of use, etc. For this reason, developers of modern software c.s. do not use already existing accumulated experience but re-invent the same or similar solutions [8], [9].
5) There are no unified universal tools for the development [10] and re-engineering of other systems that allow not only to automate their design, but also to minimize the development process itself by way of unifying the representation models of these systems and having a semantically powerful complex library of reusable components.
6) Even highly specialized software c.s. must have a good level of intelligence and a good level of trainability to solve more complex problems. Next-generation software c.s., unlike modern c.s., must operate on the meaning of what they know and process: they must understand each other [11]. [12], find common ground and form teams to solve problems of any class [13], [14].

7) Modern computers are poorly adapted for effective implementation of even existing models of knowledge representation and models for solving problems that are hard to formalize, which requires the development of fundamentally new platforms and computers that ensure the unification of the representation of this knowledge (!) [15], [16].

Typically, systems of this kind are designed and developed to solve narrow applied problems. As a result, systems with the problems described above are created. So, most of the described problems, unfortunately, were not solved when creating CADs described in the following papers [17], [18], [19], [20], [21], [22].

When designing next-generation c.s., first of all, it is necessary to take into account the shortcomings of modern c.s. This means that the design of next-generation c.s. should be reduced to solving the problems that exist in modern c.s. Thus, the following possible solutions can be identified:

1) Implementation of next-generation c.s. should not be inferior to the implementation of modern c.s. Such a task is reduced to the choice of means for storing, representing and processing data in these systems. But still, when speaking of next-generation c.s., it is necessary to lean towards a higher form of data – knowledge [14]. They must be unified in their representation, semantically integral, connected, unambiguous, etc. Thus, next-generation c.s. should be organized in such a way that the form of representation of different types of knowledge is the same (!). And this, in turn, means that the design and development tools for other systems should be organized in such a way that these systems can be easily integrated with each other, striving to increase their degree of convergence [6].
2) Since modern c.s. are platform-dependent, which in turn complicates the development of such systems, it is necessary to create such tools that allow you to create c.s. independently (!) of the implementation of these tools. At the same time, this should be done in such a way that the process of designing, developing, documenting and using such systems is carried out using the same tools and methods [6], [23], which are part of these tools, and in such a way that the quality of such systems is determined by the degree of their deep integration with each other. This can be solved with the help of general ontologies for the development of such systems, that is, with the help of an ontological [24] and component [25] approaches (!) not only to their design, but also to their implementation.

Specific solutions will be discussed and described below. This work develops the ideas and solves the problems described in the previous work [26].

III. Approaches to designing automation systems used in c.s. design and implementation

Implementation of data storages used in the vast majority of c.s. is based on the relational data model. Examples of such systems for processing unstructured and semi-structured data are the SMILA platform (SeMantic Information Logistic Architecture), Teradata Aster Discovery Platform, which implement relational, columnar and hybrid models for storing records in a database with a massively parallel MPP architecture [27], CYC platform [28], Semantic Web tools [29].

However, information systems are currently undergoing intensive intellectualization. First of all, this is due to an increase in the level of complexity of the tasks being solved. The intellectualization of information systems, like any technology for developing software systems, requires taking into account weakly formalized, possibly not completely defined, fuzzy, temporal, spatially distributed information and, as a result, obtaining structured, semi-structured and unstructured data [30]. The increase in the number of intellectual tasks of processing large amounts of data in all spheres of human activity leads to the need to create universal means of storing, presenting and processing multistructured information.

The presence of such tasks stimulates the transition from conventional databases to graph counterparts. This is explained not so much by the efficiency of memory organization and data processing in graph databases, but by the importance of representing the configurations of relationships (i.e., meaning) between them [31]. A detailed explanation of the principles of organizing graph data in databases can be found in the work of the authors of the popular Neo4j graph DBMS [32].

For a general understanding of the whole problem associated with the representation and processing of data and knowledge, we will consider several modern implementations of graph data models in the form of software products. Any of the databases described below is designed for convenient storage and access to data presented in the form of super-large graphs. According to the organization of memory and the process of data processing, graph databases can be classified into the following types:

1) databases with local storage and processing of graphs (Neo4j, HyperGraphDB, AllegroGraph);
2) databases with distributed data storage and processing (Horton, InfiniteGraph);
3) databases in "key-value" format (Trinity, CloudGraph, RedisGraph, VertexDB);
4) document-oriented databases (OrientDB);
5) add-ons for SQL-oriented databases (Filament, G-store);
6) graph databases with MapReduce model (Pregel, Apache Giraph, GraphLab).

A detailed description of each of the presented databases can be found in the works devoted to comparing relational and graph databases [33], [34], [35], [36].

The motivation for moving from conventional graph databases is due to the advantages of organizing a memory model and processing in them:

1) Data processing performance improves by one or more orders of magnitude when data is represented as graph structures, which is explained by the properties of the graph itself. Unlike relational databases, where query performance degrades as the dataset grows with increasing query intensity, graph database performance tends to remain relatively constant even as the dataset grows. This is due to the fact that data processing is localized in some part of the graph. As a result, the execution time of each request is only proportional to the size of the part of the graph traversed to satisfy this request, and not to the size of the entire graph [37].
2) Graph structures have tremendous expressive power. Graph databases offer an extremely flexible data model and way of representing [38], [39]. Graph structures are additive, which provides the flexibility to add new data relationships, new nodes, and new subgraphs to an existing structure without violating its integrity and connectivity.

As mentioned earlier, next-generation c.s., by virtue of their properties, must operate not just with data, but with knowledge. In order to understand the meaning of knowledge, it is necessary to present this knowledge in an understandable form for everyone: both for a person and for a machine. Speaking about the unification of the representation of all types of knowledge, it is important to use graph databases not just as a means for storing structured data, but for storing semantically holistic and interconnected knowledge [40]. In the context of designing next-generation c.s., we will talk about knowledge bases designed according to the principles of graph databases.

It should also be noted that the emphasis in this work is on the development of c.s. support for the design of other c.s., and for the development of complex tools to support the automatic design of next-generation intelligent computer systems, which are knowledge-driven. Such tools can be compared with knowledge base management systems [41], [42], [43], [44].

IV. Suggested solution

Despite the vast variety of classical technologies used by mankind, there is no general solution that allows solving the problem in a complex manner. At the moment, the described problems can only be solved with the help of a general and universal solution – the OSTIS Technology. The OSTIS Technology is based on a unified version of information encoding based on semantic networks with a basic set-theoretic interpretation, called SC-code. The language of semantic representation of knowledge is based on two formalisms of discrete mathematics: set theory – determines the semantics of the language – and graph theory – determines the syntax of the language. Any types and models of knowledge can be described using SC-code [6].

The platform for interpreting the semantic models of ostis-systems will simply be called the ostis-platform, which denotes the interpreter of the logical-semantic models of ostis-systems. The logical-semantic model (sc-model) of the osits-system is a formal model (formal description) of the functioning of this osits -system, consisting of (1) a formal model of information stored in the memory of the osits-system and (2) a formal model of a team of agents that process the specified information. The sc-model of the ostis-system includes the sc-memory, the sc-model of the knowledge base, the sc-model of the problem solver, and the sc-model of the interface. Each ostis-system designed using the OSTIS Technology must include a platform for interpreting the semantic models of ostis-systems (in a particular case, sc-memory) and a logical-semantic model of the ostis-system, represented using SC-code (sc-model of the ostis-system) [6], [23].

For the convenience of knowledge representation, there are three external knowledge representation languages based on the SC-code:

1) SCg-code with which knowledge is displayed in the form of graph structures understandable to the average user;
2) SCs-code in which knowledge is represented as a linear text;
3) SCn-code for displaying sc-constructions as hypertext. This representation is close to natural, understandable to the average user [6].

No other classical technology used by the engineers of the modern information society has a clearly defined, strict, formal conceptual system that could be used to solve various types of problems, not to mention the means necessary to solve the tasks set in the direction of increasing efficiency, interoperability, semantic compatibility of systems developed on the basis of these technologies. The OSTIS technology provides all the necessary capabilities and tools for developing next-generation ostis-platforms that provide efficient and high-quality interpretation of logical-semantic models of semantically compatible interoperable ostis-systems [6], [12].

The ostis-platform also means a family of software-based semantic associative computer emulators [6].

Platforms for developing other c.s. (not necessarily intelligent) should provide:

- determinism and uniqueness of interpretation of systems built on the basis of such ostis-platforms,
- availability of common language tools for formal description of designed components at different

levels of detail [45], [46],

- clear separation of the process of developing formal descriptions of components and the process of their implementation according to given formal descriptions [47], [48],
- creation and use of powerful and accessible libraries of formal descriptions of reusable ostis-platforms components,
- quality and a high level of applicability of reusable ostis-platform components.

The need for such properties in implemented ostis-platforms is justified by their purpose. Computer complexes that allow the development of other c.s. must be implemented and described in such a way that any intelligent computer system implemented on it is compatible with another similar system, so that the future interpretation of its logical-semantic model remains correct, unambiguous and independent of the means and solutions by which the ostis-platform is implemented. From this point of view, the implemented ostis-platform is only a means of mass creation of other systems and can be easily replaced by its functionally equivalent analogue that meets all the requirements for platforms [23].

In general, next-generation platforms should provide platform independence (!) of the logical-semantic models of ostis-systems implemented and interpreted on them. Different options for implementing the ostis-platform should not affect the process and result of designing ostis-systems, that is, the process and result of building logical-semantic models of the developed ostis-systems. That is, the designed ostis-systems should not depend on the specific platform for their interpretation. Thus, the platform independence of systems from specific ostis-platforms means the functional completeness of these platforms for creating systems, the simplicity and flexibility of expanding their functionality and the range of tasks to be solved, and, ultimately, the level of high intelligence of computer systems implemented on them.

The implementations of the ostis-platform designed using the OSTIS Technology *Implementations* should be based on the following fundamental principles:

- All texts represented in SC-code are graph constructions. Therefore, the task of developing a *Software implementation of the ostis-platform* is reduced to developing means for storing and processing such graph structures. In other words, the future platform should provide functionally complete and unambiguous interpretation of stored graph constructions.
- Designing a platform for interpreting sc-models of computer systems, including its components, must be clearly specified and formulated in terms of models, methods and tools for describing complex systems offered by the OSTIS Technology. It is the ontological approach to the design, operation and re-engineering of such a subclass of computer systems that will make it possible to effectively and universally develop other ostis-systems for various purposes [49], [50].

One of the ways to test, develop, and, in some cases, introduce new models and technologies, regardless of the availability of appropriate hardware, is the development of software models of this hardware that would be functionally equivalent to the hardware itself, but at the same time interpreted on the basis of traditional hardware architecture (in this paper, we will consider the von Neumann architecture as the dominant traditional architecture now). Obviously, the performance of such software models in the general case will be lower than the hardware solutions themselves, but in most cases it turns out to be sufficient to develop the corresponding technology in parallel with the development of hardware and to gradually transfer existing systems from a software model to hardware.

The popularity and development of graph databases leads to the fact that, at first glance, it seems expedient and effective to implement the *Software implementation of the ostis-platform* based on one of these tools. However, there are a number of reasons why this is not possible. These include the following:

- To ensure the efficiency of storage and processing of information structures of a certain type (in this case, SC-code structures, sc-constructions), the specificity of these structures should be taken into account. In particular, the experiments described in [51] showed a significant increase in the efficiency of their own solution compared to those existing at that time.
- Unlike classical graph constructions, where an arc or an edge can only be incident to a graph vertex (this is also true for rdf-graphs), in SC-code, it is quite typical that an sc-connector is incident to another sc-connector or even two sc-connectors. In this regard, the existing means of storing graph constructions do not allow explicit storage of sc-constructions (sc-graphs). This problem can also be solved by passing from an undirected graph to a digraph [52].
- Information processing within the framework of the OSTIS Technology is based on a multi-agent approach [53], within which agents for processing information stored in sc-memory (sc-agents) respond to events occurring in sc-memory and exchange information by specifying the actions they perform in sc-memory [54]. In this regard, one of the most important tasks is the implementation within the *Software implementation of the ostis-platform* of the possibility of subscribing to events occurring in *Implementation the sc-memory*, which at the moment is practically not supported within modern tools for storing and processing graphs structures.
- SC-code also allows describing external information

structures of any kind (images, text files, audio and video files, etc.), which are formally treated as the contents of *sc-elements*, which are signs of *external files of ostis-systems*. Thus, the *Software implementation of the ostis-platform* component should be a file memory implementation that allows storing the indicated constructions in any generally accepted formats. The implementation of such a component within the framework of modern means of storing and processing graph structures is also not always possible.

Due to the combination of the above reasons, it was decided to implement the *Software implementation of the ostis-platform* "from scratch", taking into account the peculiarities of storing and processing information within the framework of the OSTIS Technology.

V. Current Software implementation of the ostis-platform

The current *Software implementation of the ostis-platform* is web-oriented, so from this point of view, each ostis-system is a website accessible online through a regular browser. This implementation option has an obvious advantage – access to the system is possible from anywhere in the world where the Internet is available, and no specialized software is required to work with the system. On the other hand, this implementation option provides the possibility of several users operating the system in parallel. The implementation is cross-platform and can be built from source on various operating systems. At the same time, the interaction between the client and server parts is organized in such a way that web-interface can be easily replaced with a desktop or mobile interface, both universal and specialized.

Software implementation of the ostis-platform
∈ *specialized ostis-platform*
∈ *web-based implementation of the ostis-platform*
  := [implementation of the platform for interpreting sc-models of computer systems that involves the interaction users with the system via the Internet]
∈ *multi-user ostis-platform implementation*
∈ *non-atomic reusable ostis-systems component*
∈ *dependent reusable ostis-systems component*
⇒ *software system decomposition**:
  {• *Implementation of the sc-memory*
  • *Implementation of the interpreter of user interface sc-models*
  • *Implementation of a basic set of platform-specific sc-agents and their common components*
  }
⇒ *component dependencies**:
  {• *Implementation of the sc-memory*
  • *Implementation of the interpreter of user interface sc-models*
  }

The core of the platform is *Implementation of the sc-memory*, which can simultaneously interact with both *Implementation of the interpreter of user interface sc-models*, and with any third-party applications using the appropriate networking languages (network protocols). From the point of view of the overall architecture *Implementation of the interpreter of user interface sc-models* acts as one of many possible external components that interact with *Implementation of the sc-memory* over the network. It is worth noting that the current version of the ostis-platform implementation is specialized, that is, it does not include the implementation of the SCP base language interpreter.

VI. General description of Implementation of the sc-memory

Within the framework of the current *Implementation of the sc-memory*, *sc-storage* is understood as a program model component that stores sc-constructions and accesses them through the program interface. In general, *sc-storage* can be implemented in different ways. In addition to *sc-storage* itself, *Implementation of the sc-memory* also includes *Implementation of the file storage*, designed to store the contents of *internal files of ostis-systems*.

Implementation of the sc-memory
:= [Implementation of the sc-machine]
⇐ *software model**:
  sc-memory
∈ *software model of the sc-memory based on linear memory*
∈ *non-atomic reusable ostis-systems component*
∈ *dependent reusable ostis-systems component*
⇒ *software system decomposition**:
  {• *Implementation of the sc-storage*
  • *Implementation of the file storage*
  • *Implementation of the subsystem of interaction with external environment using networking languages*
  • *Implementation of auxiliary tools for working with sc-memory*
  }
⇒ *component dependencies**:
  {• *GLib library of methods and data structures*
  • *C++ Standard Library for Methods and Data Structures*
  • *Implementation of sc-storage*
  • *File storage implementation*
  }

It should be noted that when switching from *Implementation of the sc-memory* to its hardware implementation, it would be advisable to implement the file memory of the ostis-system based on traditional linear memory (at least at the first stages of *semantic computer* development). The current version of *Implementation of the sc-memory* is open and available at [55].

Within this *Implementation of the sc-storage*, *sc-memory* is modeled as a set of *segments*, each of which is a fixed-size ordered sequence of *sc-storage* elements, each of which corresponds to specific sc-element. Currently, each segment consists of $2^{16} - 1 = 65535$ *sc-storage elements*. Each segment consists of a set of data structures describing specific *sc-elements* (sc-storage elements). Regardless of the type of sc-element being described, each *sc-storage element* has a fixed size (currently 36 bytes), which ensures convenient storage. Thus, the maximum size of the knowledge base in the current sc-memory software model can reach 180 GB (excluding the contents of *internal files of the ostis-system* stored on the external file system).

VII. Implementation of the sc-storage

A. Selected solution and its rationale

Implementation of the sc-storage must meet the following requirements:

- high performance – minimizing the time spent on adding, deleting and accessing stored information;
- minimal memory and disk space for storing sc-texts;
- scalability – the ability to easily add computing power as the load increases.

The sc-storage consists of sc-segments of elements that correspond to some sc-elements of the abstract SC-code. Each segment of the sc-storage has a number relative to the sc-storage itself, and each element of some sc-storage sc-segment has a number relative to that sc-segment.

Allocation of *sc-storage segments* makes it possible, on the one hand, to simplify address access to *sc-storage elements*, and on the other hand, to realize the possibility of unloading a part of sc-memory from RAM to the file system if necessary. In the second case, the sc-storage segment becomes the minimum (atomic) paged part of the sc-memory. The segment unloading mechanism is implemented in accordance with the existing principles of virtual memory organization in modern operating systems.

The maximum possible number of segments is limited by the settings of the software implementation of the sc-storage (currently the number of sc-segments is $2^{16} - 1 = 65535$ by default, but in the general case it may be different). Thus, technically, the maximum number of stored sc-elements in the current implementation is about 4.3×10^9 sc-elements. By default, all segments are physically located in RAM, if there is not enough memory, then a mechanism is provided for unloading some of the sc-segments to the hard disk (virtual memory mechanism).

The current version of the *Implementation of the sc-memory* assumes the possibility of saving the memory state (imprint) to the hard disk and subsequent loading from the previously saved state. This feature is necessary to restart the system in case of possible failures, as well as when working with the source code of the knowledge base, when the assembly from the source code is reduced to the formation of a snapshot of the memory state, which is then placed in the *Implementation of the sc-memory*.

B. General description of the current implementation of sc-storage

Implementation of the sc-storage
$\in$ *implementation of sc-storage based on linear memory*
$\in$ *non-atomic reusable ostis-systems component*
$\in$ *dependent reusable ostis-systems component*
$\Leftarrow$ *software model**:
  sc-storage
    $\Leftarrow$ *subset family**:
      sc-storage segment
        := [sc-storage page]
        $\Leftarrow$ *subset family**:
          sc-storage element
$\Rightarrow$ *component dependencies**:
  {• *GLib library of methods and data structures*
   • *C++ Standard Library of Methods and Data Structures*
  }
$\Rightarrow$ *used method representation language**:
  • *C*
  • *C++*
$\Rightarrow$ *internal language**:
  • *SCin-code*

Each sc-storage element in the current implementation can be uniquely specified by its address (sc-address), which consists of the sc-segment number and the *sc-storage element* number within the sc-segment. Thus, the *sc-address* serves as the unique coordinates of an *sc-storage element* within the framework of the *Implementation of the sc-storage*.

For each sc-address, it is possible to assign one-to-one correspondence to some hash obtained as a result of applying a special hash function on this sc-address. The hash is a non-negative integer and is the result of converting the number of the sc-storage segment si, in which the sc-element is located, and the number of this sc-element of the sc-storage *ei* within this sc-segment *si*. The sc-storage framework uses a single hash function to get the hash of the sc-address of the sc-element and is specified as $f(si, ei) = si << 16 \vee ei \wedge 0xffff$, where the operation $<<$ is the operation logical bit shift left of the left argument by the number of units specified by the right argument, relative to of this operation, the $\vee$ operation is a logical *OR* operation, the $\wedge$ operation is a logical *AND* operation, the number $0xffff$ is the number 65535, represented in hexadecimal form and denoting the maximum number of sc-elements in one sc-storage segment.

sc-address
:= [address of the sc-storage element corresponding to the given sc-element within the current implementation of the sc-storage as part of software model of sc-memory]
∈ *32-bit integer*

The sc-address is not taken into account in any way when processing the knowledge base at the semantic level and is only necessary to provide access to the corresponding data structure stored in linear memory at the *Implementation of the sc-storage* level. In general, sc-address of the sc-storage element corresponding to the given sc-element may change, for example, when rebuilding the knowledge base from source texts and then restarting the system. At the same time, the sc-address of the sc-storage element corresponding to the given sc-element cannot change directly during the system operation in the current implementation. For simplicity, we will say "sc-address of the sc-element", meaning *sc-address* of the *sc-storage element* that uniquely corresponds to the given *sc-element*.

The specification of such complex software objects must be represented in some kind of knowledge representation language, in this case SC-code. From the point of view of SC-code itself, the language that should describe the *Software implementation of the ostis-platform* is a sublanguage of SC-code, that is, it inherits all the properties of the syntax and denotational semantics of SC-code, and a metalanguage for describing the representation of the SC-code constructions in the memory of a software emulator of a semantic associative computer. Such a model for presenting the specification of a c.s., which is a platform for the creation, use and development of other c.s., certainly provides strong advantages over other options for presenting c.s. specifications:

1) The language, the texts of which the system stores and processes, and the language of the specification of how the system represents the texts of the first language in the memory of itself, are subsets of the same language. This simplifies not only the understanding of a developer who develops a complex software system, due to the fact that the form of representation of the language processed by this system and the language of its specification is unified, but also allows you to open new functionality for this system in knowing itself. Thus, this approach makes it possible to fully realize the properties of an intelligent system, for example, reflexivity.
2) It is impossible to design and implement intelligent c.s. on a computer system that is not itself intelligent. Presenting the specification of a system in this form makes it possible to increase the level of its intelligence.
3) Since the form of representation of the language describing the system is unified with the language it processes, there is no need to create additional tools for verification and analysis of the system operation.

C. The concept of SCin-code

We will call such a language the language of the internal representation of SC-code, or, briefly, *SCin-code (Semantic Code interior)*. Sc-storage of SC-code texts can be considered as a subset of scin text.

SCin-code
:= [Semantic Code interior]
:= [Language of the internal semantic representation of the SC-code inside the memory of the ostis-system]
:= [meta-language for describing the representation of the SC-code inside the memory of the ostis-system]
⇒ *frequently used non-primary sc-element external identifier**:
  [scin-text]
    ∈ *common noun*
∈ *abstract language*
∈ *metalanguage*
⊂ *SC-code*
⊃ *sc-storage*

SCin-code syntax is given by: (1) *SCin-code alphabet*, (2) one-to-one correspondence *sc-addresses**.

D. SCin-code alphabet

SCin-code alphabet^
:= [syntactic type of sc-storage element]
:= [Set of types of sc-storage elements]
⇐ *alphabet**:
  SCin-code
= {• *sc-storage element corresponding to sc-node*
  • *sc-storage element corresponding to sc-arc*
  • *sc-storage element with null sc-address*
    ∈ *singleton*
  }

*SCin-code alphabet^* consists of three syntactically distinguished types of sc-storage elements: an sc-storage element corresponding to a general sc-node, an sc-storage element corresponding to a general sc-arc, and an sc-storage element, having a null sc-address. Such an alphabet not only allows you to set in memory the minimum set of objects with which you can perform computational operations, but, if necessary, is convenient for expansion. So, for example, the given alphabet of the language can be extended by adding to it *sc-storage element, corresponding to the internal file of ostis-system* or *sc-storage element, corresponding to sc-edge*.

sc-storage element corresponding to sc-element
∈ *sc-element*
:= [sc-storage element]
:= [sc-storage cell]
:= [sc-element image within sc-storage]
:= [data structure, each instance of which within sc-storage corresponds to one sc-element]
⇒ *subdividing**:
*SCin-code alphabet^*

The relation *sc-address** is defined as a one-to-one correspondence, the first component of each ordered pair of which is some element of the sc-storage corresponding to some sc-element, and the second component is the sc-address of this element of the sc-storage.

sc-address*
∈ *one-to-one correspondence*
⇒ *first domain**:
sc-storage element correspoinding sc-element
⇒ *second domain**:
16-bit integer

E. Syntax and syntactic rules of SCin-code

Within *Implementation of sc-storage* there must be a set of *syntactic and semantic classes of sc-storage elements* that:

1) define the element type at the platform level and does not have a corresponding sc-arc of membership (more precisely, a base sc-arc) explicitly stored in sc-memory (its presence is implied, but it is not explicitly stored, since it will lead to infinite increasing the number of sc-elements to be stored in sc-memory);
2) can be represented as parameters of the corresponding elements of the sc-storage, that is, a set of such elements, each of which has a "label" expressed by some numerical value;
3) can specify the type of elements of the sc-storage with the level of detail that is necessary so that, for example, when performing a search operation using such element classes, it is easy to determine the class of a particular element.

For this purpose, the basic syntactic classification of its elements is allocated in the SCin-code. In order to represent and store any constructions of the SC-code, it is enough to have only two base classes of sc-storage elements, while the remaining classes of sc-storage elements can be added in the extended version of the SCin-code and thereby implement the necessary logic at the level of sc-memory Implementation .

Syntactic classification of SCin-code elements
⊃=
{

sc-storage element corresponding to sc-element
⇒ *subdividing**:
{• *sc-storage element corresponding to sc-node*
• *sc-storage element corresponding to sc-arc*
}
}

It should be noted that all classes of sc-storage elements that are part of the syntactic classification of SCin-code elements are syntactically distinguished classes of SCin-code elements.

Although the *sc-addresses** relation makes it possible to completely describe the links between the elements of the sc-storage of the ostis-system, but for the specification of the representation of SC-code constructions inside the memory of the ostis-system, only one *sc-address** relation is not always enough to indicate completely exactly and clearly the relationships between the elements of the sc-storage corresponding to the sc-elements of these constructions. Therefore, in practice, when describing the representation of SC-code structures inside the memory of the ostis-system, it is necessary to use more particular relations of this basic relation, for example, such as *sc-address of the sc-storage element corresponding to the outgoing sc-arc from the given sc-element **, *sc-address of the sc-storage element corresponding to the incoming sc-arc in the given sc-element**, *sc-address of the sc-storage element corresponding to the incident sc-element of the sc-arc**.

sc-address*
⇒ *subdivinding**:
{• *sc-address of the sc-storage element corresponding to the outgoing sc-arc from the given sc-element**
• *sc-address of the sc-storage element corresponding to the incoming sc-arc in the given sc-element**
• *sc-address of the sc-storage element*

*corresponding to the incident sc-element of the sc-arc**
}

The *sc-address of the sc-storage element corresponding to the outgoing sc-arc from the given sc-element** is defined as a binary oriented relation, the first component of each oriented pair of which is some element of the sc-storage corresponding to some sc-element from which the given sc-arc comes out, and the second component is the sc-address of this outgoing sc-arc. Particular types of this relation are the relation *sc-address of the sc-storage element corresponding to the initial outgoing sc-arc from the given sc-element**, the relation *sc-address of the sc-storage element corresponding to the next outgoing sc-arc from of the given sc-element** and the relation *sc-address of the sc-storage element corresponding to the previous outgoing sc-arc from the given sc-element**.

sc-address of the sc-storage element corresponding to the outgoing sc-arc from the given sc-element*
⇒ *subdividing**:
{
- *sc-address of the sc-storage element corresponding to the initial outgoing sc-arc from the given sc-element**
- *sc-address of the sc-storage element corresponding to the next outgoing sc-arc from the given sc-element**
- *sc-address of the sc-storage element corresponding to the previous outgoing sc-arc from the given sc-element**

}

The relation *sc-address of the sc-storage element corresponding to the incoming sc-arc in the given sc-element** is defined as a binary oriented relation, the first component of each oriented pair of which is some element of the sc-storage corresponding to some sc-element, in which this sc-arc enters, and the second component is the sc-address of this incoming sc-arc. Particular types of this relation are the relation *sc-address of the sc-storage element corresponding to the initial incoming sc-arc in the given sc-element**, the relation *sc-address of the sc-storage element corresponding to the next incoming sc-arc in the given sc-element** and the relation *sc-address of the sc-storage element corresponding to the previous incoming sc-arc in the given sc-element**.

sc-address of the sc-storage element corresponding to the incoming sc-arc in the given sc-element*
⇒ *subdividing**:
{
- *sc-address of the sc-storage element corresponding to the initial incoming sc-arc in the given sc-element**
- *sc-address of the sc-storage element corresponding to the next incoming sc-arc in the given sc-element**
- *sc-address of the sc-storage element corresponding to the previous incoming sc-arc in the given sc-element**

}

The relation *sc-address of the sc-storage element corresponding to the incident sc-element of the sc-arc** is defined as a binary oriented relation, the first component of each oriented pair of which is some element of the sc-storage corresponding to some sc-element, which is sc- arc, and the second component is the sc-address of some sc-element incident to it. Particular types of this relation are the relation *sc-address of the sc-storage element corresponding to the initial sc-element of the sc-arc** and the relation *sc-address of the sc-storage element corresponding to the final sc-element of the sc-arc** .

sc-address of the sc-repository element corresponding to the incident sc-element of the sc-arc*
⇒ *subdividing**:
{
- *sc-address of the sc-storage element corresponding to the initial sc-element of the sc-arc**
- *sc-address of the sc-storage element corresponding to the final sc-element of the sc-arc**

}

The following restrictions are imposed on the syntactic constructions of the SCin code:

- Each *sc-storage element corresponding to an sc-element*, has a one-to-one relation to its sc-address.
- For each *sc-storage element corresponding to the sc-node*, there is one and only one relation pair *sc-addresses of the sc-storage element corresponding to the initial outgoing sc-arc from the given sc-element** and one and only one relation pair *sc-addresses of the sc-store element corresponding to the initial incoming sc-arc in the given sc-element**.
- For each *sc-storage element corresponding to the outgoing sc-arc from the given sc-element* (*sc-storage element corresponding to the incoming sc-arc to the given sc-element*), there is at most one relation pair *sc-addresses of the sc-storage element corresponding to the next outgoing sc-arc from the given sc-element** (*sc-addresses of the sc-storage element corresponding to the next incoming sc-arc in the given sc-element**) and at most one relation pair *sc-addresses of the sc-storage element corresponding to the previous outgoing sc-arc from the given sc-element** (*sc-addresses of the sc-storage element corresponding to the previous incoming sc-arc to the given sc-element**).
- For each *sc-storage element corresponding to the*

sc-arc that is the second component of each pair of the *sc-address relation of the sc-store element corresponding to the initial outgoing sc-arc from the given sc-element** (*sc-addresses of the sc-storage element corresponding to the initial incoming sc-arc in the given sc-element**) there is only one pair *sc-addresses of the sc-storage element corresponding to the next outgoing sc-arc from the given sc-element** (*sc-addresses of the sc-storage element corresponding to the next incoming sc-arc in the given sc-element**).

F. Denotational semantics of SCin-code

According to the above, for each class of sc-elements of the SC-code, there must be a program model of the class of sc-store elements that satisfies all the listed requirements. Therefore, it is important that *SCin-code Alphabet* is initially complete in order to immerse not only sc-constructions *SC-code Core*, but also its extended version. For this, semantic classes of sc-storage elements have been developed, the specification of which is represented as *Semantic classification of SCin-code elements*.

Semantic classification of SCin-code elements
⊃=
{

sc-storage element corresponding to sc-element
⇒ *subdividing**:
Typology of sc-storage elements based on constantness^
= {• *sc-storage element corresponding to sc-constant*
• *sc-storage element corresponding to sc-variable*
• *sc-storage element corresponding to sc-meta-variable*
}
⇒ *subdividing**:
Typology of sc-storage elements based on permanency^
= {• *sc-storage element corresponding to permanent sc-element*
• *sc-storage element corresponding to temporary sc-element*
}
⇒ *subdividing**:
Typology of sc-storage elements based on accessibility^
:= [sc-storage element access level class]
= {• *sc-storage element corresponding to sc-element on which read access is allowed*
• *sc-storage element corresponding to sc-element on which write access is allowed*
}
⇒ *include**:
sc-storage element corresponding to internal ostis-system file

sc-storage element corresponding to generic sc-node
⇒ *subdividing**:
Structural typology of sc-storage elements corresponding to sc-nodes^
= {• *sc-storage element corresponding to sc-node denoting a non-binary sc-link*
• *sc-storage element corresponding to sc-class*
• *sc-storage element corresponding to sc-node denoting a class of classes*
• *sc-storage element corresponding to sc-structure*
• *sc-storage element corresponding to sc-node denoting the role relation*
• *sc-storage element corresponding to sc-node denoting a non-role relation*
• *sc-storage element corresponding to sc-node denoting the primary entity*
}
⇒ *subdividing**:
Structural typology of sc-storage elements corresponding to sc-arcs^
= {• *sc-storage element corresponding to sc-arc of membership*
• *sc-storage element corresponding to generic sc-arc*
}
⇒ *subdividing**:
Typology of sc-storage elements corresponding to sc-arcs of membership, according to the type of denoted membership^
= {• *sc-storage element corresponding to sc-arc of positive membership*
• *sc-storage element corresponding to sc-arc of fuzzy membership*
• *sc-storage element corresponding to sc-arc of negative membership*
}
}

All semantically and syntactically distinguished classes of sc-storage elements, as well as all possible subclasses of these classes, are instances (elements) of the class.

At the moment, sc-edges are stored in the same way as sc-arcs, that is, they have a start and end sc-element, the difference is only in the *sc-storage element syntactic type*. This leads to a number of inconveniences during processing, but sc-edges are currently used quite rarely.

G. SCin-code specification

The specification of a SCin-code is the union of the specification of its elements. For each element, links between elements and their properties, restrictions are imposed in the form of syntactic rules described above.

sc-storage element corresponding to sc-element
∈ *sc-storage element syntactic type*
⇒ *specification**:
{}
⊃ *relation narrowing by the first domain (sign specification*, sc-storage element corresponding to sc-node)**
⊃ *relation narrowing by the first domain (sign specification*, sc-storage element corresponding to sc-arc)**

sc-storage element corresponding to sc-node
⇒ *specification**:
{• *class of sc-storage element corresponding to sc-node*
• *sc-storage element access level class*
• *sc-address**
• *sc-address of the first sc-arc outgoing from the given sc-element**
• *sc-address of the first sc-arc incoming in the given sc-element**
}

sc-storage element corresponding to sc-arc
⇒ *specification**:
{• *class of sc-storage element corresponding to sc-arc*
• *sc-storage element access level class*
• *sc-address**
• *sc-address of the sc-storage element corresponding to the initial sc-element of the sc-arc**
• *sc-address of the sc-storage element corresponding to the final sc-element of the sc-arc**
• *sc-address of the sc-storage element corresponding to the initial outgoing sc-arc from the given sc-element**
• *sc-address of the sc-storage element corresponding to the initial incoming sc-arc in the given sc-element**
• *sc-address of the sc-storage element corresponding to the next outgoing sc-arc from the given sc-element**
• *sc-address of the sc-storage element corresponding to the next incoming sc-arc in the given sc-element**
• *sc-address of the sc-storage element corresponding to the previous outgoing sc-arc from the given sc-element**
• *sc-address of the sc-storage element corresponding to the previous incoming sc-arc in the given sc-element**
}

H. General algorithm for embedding the SC-code construction into the memory of an ostis-system

Loading an sc-construction into the memory of the ostis-system means translating each sc-element of this sc-construction and the incidence relations between these sc-elements into the memory of the ostis-system, i.e. translating the syntactic structure of the sc-construction into the corresponding representation inside the memory of the ostis-system. In the general case, the algorithm for loading any arbitrary sc-construction into the memory of the ostis-system consists of the following steps:

1) Selection of sc-nodes and internal files of the sc-construction and saving to the corresponding memory cells of the ostis-system;
2) Select all free sc-connectors (i.e. sc-connectors whose start and end sc-element is not another sc-connector), store all sc-connectors in the corresponding ostis-system memory cells and establish links between the initial and final sc-elements of these sc-connectors;
3) Return to step 2 if there are unloaded sc-connectors;
4) Loading the contents of all internal files of the ostis-system into its file storage.

I. Example of the specification of the representation of the SC-code construction in the memory of the ostis-system

The figure 1 shows an example of the specification of the representation of an sc-construction in the memory of an ostis-system implemented on the basis of the designed ostis-platform. Here, each sc-element of the given sc-construction is assigned an sc-element denoting the storage element. For each sc-element denoting a storage element of some sc-element of a given sc-construction, its own denotational semantics is described: links between sc-storage elements and syntactic and semantic classes of elements.

J. Advantages and disadvantages of SCin-code

This model of representation in memory of a system of syntactic and semantic classes of sc-elements in the form of syntactic and semantic classes of elements of

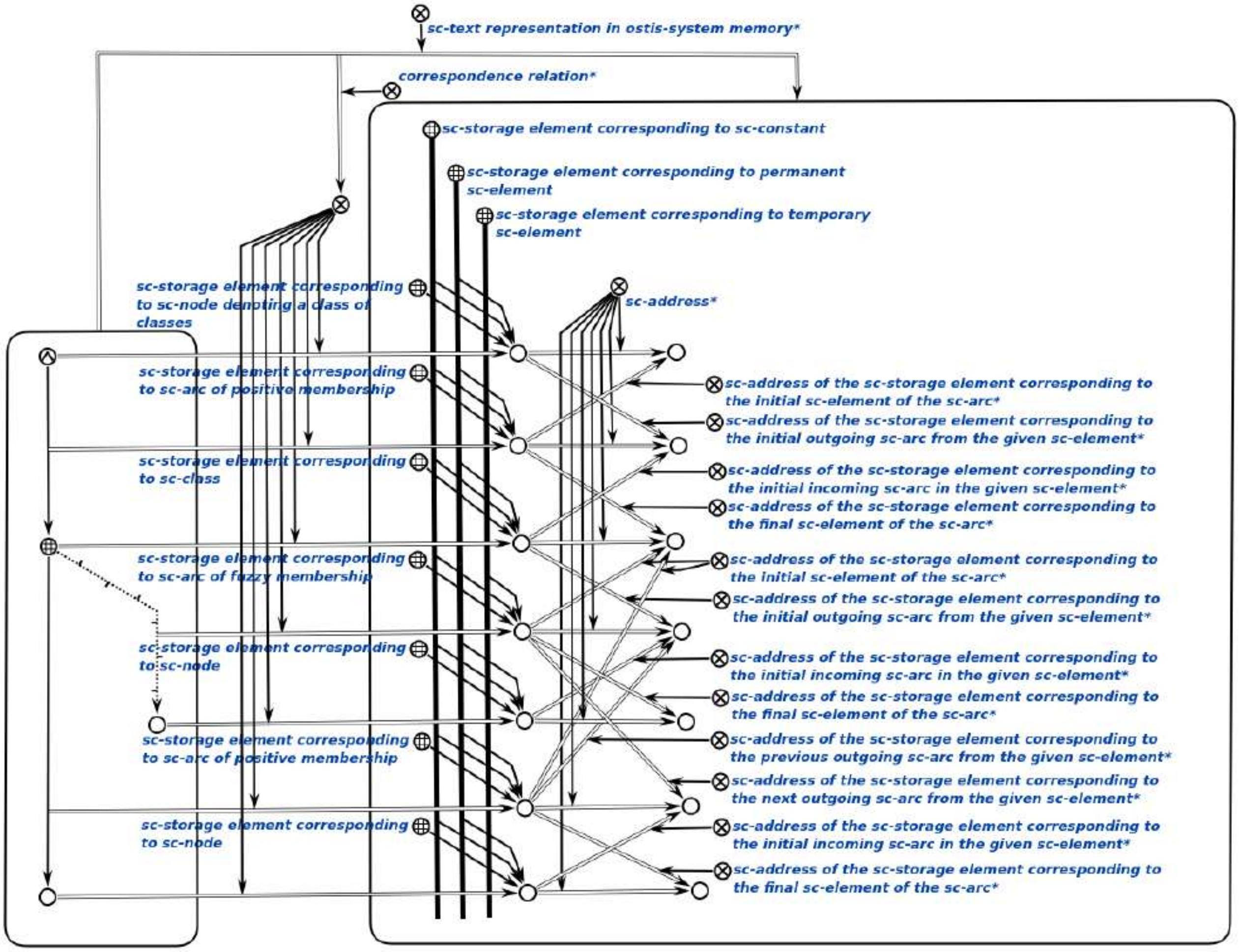

sc-text representation in ostis-system memory*
correspondence relation*
sc-storage element corresponding to sc-constant
sc-storage element corresponding to permanent sc-element
sc-storage element corresponding to temporary sc-element
sc-storage element corresponding to sc-node denoting a class of classes
sc-storage element corresponding to sc-arc of positive membership
sc-storage element corresponding to sc-class
sc-storage element corresponding to sc-arc of fuzzy membership
sc-storage element corresponding to sc-node
sc-storage element corresponding to sc-arc of positive membership
sc-storage element corresponding to sc-node
sc-address*
sc-address of the sc-storage element corresponding to the initial sc-element of the sc-arc*
sc-address of the sc-storage element corresponding to the initial outgoing sc-arc from the given sc-element*
sc-address of the sc-storage element corresponding to the initial incoming sc-arc in the given sc-element*
sc-address of the sc-storage element corresponding to the final sc-element of the sc-arc*
sc-address of the sc-storage element corresponding to the initial sc-element of the sc-arc*
sc-address of the sc-storage element corresponding to the initial outgoing sc-arc from the given sc-element*
sc-address of the sc-storage element corresponding to the initial incoming sc-arc in the given sc-element*
sc-address of the sc-storage element corresponding to the final sc-element of the sc-arc*
sc-address of the sc-storage element corresponding to the previous outgoing sc-arc from the given sc-element*
sc-address of the sc-storage element corresponding to the next outgoing sc-arc from the given sc-element*
sc-address of the sc-storage element corresponding to the initial incoming sc-arc in the given sc-element*
sc-address of the sc-storage element corresponding to the final sc-element of the sc-arc*

Figure 1. Example of the specification of the representation of the SC-code construction in the memory of the ostis-system

storage sc-elements that correspond to the first ones has a number of advantages:

- *Syntactic and semantic classes of sc-storage elements* can be combined with each other to obtain more specific classes. From the point of view of software implementation, such a combination is expressed by the operation of bit-wise addition of the values of the corresponding numerical expressions *classes of elements of the sc-storage* (here, in the specification on the SC-code, this can be done using the intersection of the corresponding classes). For example, bit-wise addition of numeric expressions of sc-storage element classes corresponding to sc-node and sc-constant results in a new sc-storage element class – *sc-storage element corresponding to constant sc-node*.
- Numeric expressions of some classes may match. This is done to reduce the size of the sc-storage element by reducing the maximum size of the numeric expression of the class of these elements. There is no conflict in this case, since such classes cannot be combined, for example *sc-storage element corresponding to the sc-node of the role relation* and *sc-storage element corresponding to the fuzzy membership sc-arc*.
- It is important to note that each of the selected classes of elements (except for classes obtained by combining other classes) uniquely corresponds to the ordinal number of a bit in linear memory, which can be seen by looking at the corresponding numerical expressions of these classes. This means that classes of elements are not included in each other (although this is not the case in the specification), for example, specifying membership in the class *of sc-store elements corresponding to an sc-arc of positive membership* does not automatically indicate the membership of *elements of the sc-storage corresponding to the sc-arc of membership*. At the implementation level, this makes label combination and comparison operations more efficient.

However, an increase in their number, although it improves the performance of the platform by simplifying some operations for checking the class of an sc-storage element, it leads to an increase in the number of situations in which it is necessary to take into account the explicit and implicit representation of sc-arcs, which, in turn, complicates the development of the platform and development of program code for processing stored sc-construction. This model does not allow sufficient representation of the syntactic and semantic classes of sc-elements, since it has the following important disadvantages:

- At the moment, the number of *syntactic classes of sc-storage elements* is large enough, which leads to a fairly large number of situations in which it is necessary to take into account the explicit and implicit storage of sc-arcs belonging to the corresponding classes. On the other hand, changing the set of classes of elements for any purpose in the current implementation is a rather laborious task (in terms of the amount of changes in the program code of the platform and sc-agents implemented at the platform level), and expanding the set of classes without increasing the volume sc-storage element in bytes turns out to be completely impossible. The solution to this problem is to minimize the number of classes as much as possible, for example, to the number of classes corresponding to the *SC-code alphabet*. In this case, the membership of sc-elements to any other classes will be recorded explicitly, and the number of situations in which it will be necessary to take into account the implicit storage of sc-arcs will be minimal.
- Some class from the current set of *syntactic and semantic classes of sc-elements* are rarely used (for example, *sc-store element corresponding to a generic sc-edge* or *sc-store element corresponding to sc-arc of negative membership*), in turn, in sc-memory there can be classes that have quite a lot of elements (for example, *binary relation** or *number*). This fact does not allow us to fully use the efficiency of having classes. The solution to this problem is the rejection of a previously known set of classes and the transition to a dynamic set of classes (while their number can remain fixed). In this case, a set of classes expressed as numeric values will be formed based on some criteria, for example, the number of elements of this class or the frequency of calls to it.
- *base sc-arcs* denoting that sc-elements belong to some known limited set of classes in memory are presented implicitly. This fact must be taken into account in a number of cases, for example, when checking whether an sc-element belongs to a certain class, when searching for all outgoing sc-arcs from a given sc-element, etc. If necessary, some of these implicitly stored sc-arcs can be represented explicitly, for example, in the case when such an sc-arc must be included in some set, that is, another sc-arc must be drawn into it. In this case, it becomes necessary to synchronize changes associated with a given sc-arc (for example, its deletion) in its explicit and implicit representation. The current *Implementation of sc-storage* does not implement this mechanism. This problem is solved by one of the previous options for solving the problems of this model.

In the current *Implementation of sc-storage access-level classes* are used to provide the ability to restrict the access of some processes in sc-memory to certain elements stored in sc-memory. Each sc-store element belongs to one of two classes: the class *of sc-store elements corresponding to sc-elements on which the read right is allowed* and the class *of sc-store elements corresponding to sc-elements on which the right is allowed records*. Each of which is expressed as a number from 0 to 255.

Thus, the null value of the numeric expressions of the class *sc-storage elements corresponding to sc-elements on which read access is allowed* and the class *sc-storage elements corresponding to sc-elements on which write access is allowed* means that any process can get unrestricted access to this sc-storage element.

Each element of the sc-storage corresponding to some sc-element is described by its syntactic type (label), and, regardless of the type, the sc-address of the first sc-arc entering the given sc-element and the first sc-arc leaving the given sc-element is indicated (may be empty if there are no such sc-arcs). The remaining bytes, depending on the type of the corresponding sc-element (sc-node or sc-arc), can be used to store the specification of the sc-arc. Also, *sc-address of the first sc-arc outgoing from the given sc-element** and *sc-address of the first sc-arc entering the given sc-element** may generally be absent (be null, "empty"), but the size of the sc-element in bytes will remain the same.

From the point of view of software implementation, the data structure for storing the sc-node and sc-arc remains the same, but the list of fields (components) changes in it. In addition, as you can see, each sc-storage element (including *sc-storage element corresponding to sc-arc*) does not store a list of sc-addresses of associated sc-elements, but stores sc-addresses of one outgoing and one incoming arc, each of which in turn stores the sc-addresses of the next and previous arcs in the list of outgoing and incoming sc-arcs for the corresponding elements. All of the above allows you to:

- make the size of such a structure fixed (currently 36 bytes) and independent of the syntactic type of the stored sc-element;
- provide the ability to work with sc-elements without regard to their syntactic type in cases where it is necessary (for example, when implementing search queries like “Which sc-elements are elements of this set”, “Which sc-elements are directly related with the given sc-element”, etc.);

- provide the ability to access *sc-storage element* in constant time;
- provide the ability to place the *sc-storage element* in the processor cache, which in turn speeds up the processing of sc-constructions;

VIII. Implementation of the ostis-system file storage

A. Selected solution and its rationale

Often, the expressiveness of SC-code graph structures is not enough to represent and store linear sequences of texts, pictures, sound, video, and so on. Although SC-code is a universal tool for representing any kind of knowledge, there is not always a need to immerse something in the graph-dynamic memory of the ostis-system, at least in the early stages of development of the ostis-platform. This can also be explained by the fact that information constructions that do not belong to the SC-code are quite complex in syntax and volume. To solve such problems, an additional element is introduced at the *SC-code Alphabet^* level – the internal file of the ostis-system. With the help of ostis-system files, it is possible to represent, store, process and visualize information structures that do not belong to the SC-code using the SC-code.

Therefore, when implementing sc-memory, it is necessary to take into account the need to store information structures that do not belong to SC-code using SC-code. Such solution is the *Implementation of the file storage of the ostis-system.*

During the entire period of *Software implementation of the ostis-platform* development, there have been quite a few attempts to implement a fully functional and fast file storage based on popular databases. However, all these solutions did not take into account potential problems in the implementation of the search and navigation subsystem *Software implementation of the ostis-platform*. Now the file storage is implemented by its own means, as data structures for storing information structures that do not belong to the SC-code, prefix B-trees [56] and linear lists are used.

The choice is justified by the fact that:

- prefix structures are fairly easy to understand and minimal in their syntax;
- with the help of prefix structures, it is quite convenient to store and process key-value relations;
- accessing a value by key occurs in the worst case for the length of that key [57], [58];
- due to the fact that the prefixes stick together, there is a strong gain in memory usage.

To store the contents of internal files of ostis-systems, files are used that are explicitly stored on the file system, which is accessed by means of the operating system on which *Software implementation of the ostis-platform* is running.

Implementation of the ostis-system file storage
∈ *file storage implementation based on prefix tree*
⇐ *software model**:
ostis-system file storage
∈ *atomic reusable ostis-systems component*
∈ *dependent reusable ostis-systems component*
⇒ *component dependencies**:
{• *GLib library of methods and data structures*
}
⇒ *used method representation language**:
• *C*
⇒ *internal language**:
• *SCfin-code*

As in the case with the sc-storage, it is necessary to describe the language for representing information structures that do not belong to the SC-code inside the file storage of the ostis-system.

B. The concept of the SCfin-code

We will call such a language the language of internal representation of information constructions that do not belong to the SC-code, or, briefly, *SCfin-code (Semantic Code file interior)*. The file storage of texts that do not belong to the SC-code can be considered as a subset of the scfin-text.

SCfin-code
:= [Semantic Code file interior]
:= [Language of the internal semantic representation of information constructions that do not belong to the SC-code inside the memory of the ostis-system]
:= [meta-language for describing the representation of the information constructions that do not belong to the SC-code inside the memory of the ostis-system]
⇒ *frequently used non-primary sc-element external identifier**:
[scfin-text]
∈ *common noun*
∈ *abstract language*
∈ *metalanguage*
⊂ *SC-code*
⊃ *ostis-system file storage*

The *SCfin-code syntax* is given by: (1) the *SCfin code alphabet*, (2) the *sequence in linear text** order relation.

C. SCfin-code alphabet

SCfin-code alphabet^
:= [syntactic type of ostis-system file storage element]
:= [Set of types of ostis-system file storage elements]

⇐ *alphabet**:
SCfin-code
= {• *element of ostis-system file storage corresponding to a substring of linear language text*
}

*SCfin-code alphabet^* consists of one syntactically distinguished type of file storage elements – *element of ostis-system file storage corresponding to a substring of linear language text.*

element of ostis-system file storage corresponding to a substring of linear language text
∈ *sc-element*
:= [ostis-system file storage element]
:= [ostis-system file storage cell]
:= [image of information construction substring that do not belong to the SC-code within the ostis-system file storage]

The relation *sequences in a linear text** is defined as a binary oriented order relation, the components of each ordered pair of which are elements of the ostis-system file storage corresponding to some substrings of the linear text, as a result of which, as a result of their concatenation, a substring belonging to the same linear text is formed.

D. SCfin-code syntax

The *SCfin-code syntax* is quite simple, since the information constructions on it are specified using the *SCfin-code alphabet*, whose cardinality is 1, and the single incidence relation *sequence in a linear text**. Hierarchies of syntactic elements are not distinguished as such, as this is not necessary.

E. Denotational semantics of SCfin-code

At the implementation level, it is important to single out the semantic classes e*elements of the ostis-system file storage, corresponding to a substring of the text of the linear language*, which denote some prefix or postfix part of the entire information construction.

Semantic classification of SCfin-code elements
⊃=
{

element of ostis-system file storage corresponding to a substring of linear language text
⇒ *subdividing**:
Typology of elements by substring location in linear text
= {• *element of the ostis-system file storage corresponding to the prefix substring of the linear language text*
• *element of the ostis-system file storage corresponding to the postfix substring of the linear language text*
}
}

F. Example of the specification of the representation of information constructions that doesn't belong to the SC-code in the memory of the ostis-system

In the SCfin-code, it is enough to simply set the information constructions of any linear texts. However, from the point of view of the implemented sc-memory model, there is a need to specify not so much the form of information structures that do not belong to the SC-code inside the file storage of the ostis-system, but rather the links between these external information structures, the files of the ostis-system, which are signs of the SC-code. At the same time, at the sc-memory level, both the method for obtaining ostis-system files that contain a given external information structure and the methods for obtaining external information structures from given ostis-system files must be implemented at the sc-memory level.

Figure 2 shows the representation of information constructions that do not belong to the SC-code and the correspondence between ostis-system files and information constructions. Using the relation *set of sc-addresses of ostis-system files by their content prefixes**, a binary oriented pair is specified, the first component of which is a prefix structure, the elements of which are substrings of external information constructions, and the second component is the set of corresponding sc-addresses ostis-system files. And using the relation *set of postfixes of the contents of ostis-system files by their sc-addresses**, a binary oriented pair is specified, the first component of which is a prefix structure, the elements of which are substrings of the sc-addresses of ostis-system files presented in string form, and the second component is the set of corresponding postfixes of external information structures of the prefix structure, which is the first component of each pair of the relation *set of sc-addresses of ostis-system files by their content prefixes**.

G. Advantages and disadvantages of SCfin-code

The used *Implementation of the ostis-system file storage* fully justifies itself when interacting with the system. Due to the use of prefix structures, the asymptotic complexities of the method for obtaining a set of external information constructions from given ostis-system files and the method for obtaining a set of ostis-system files from given external information constructions are linear, since it depends on the length of a given string and the structure of the prefix tree.

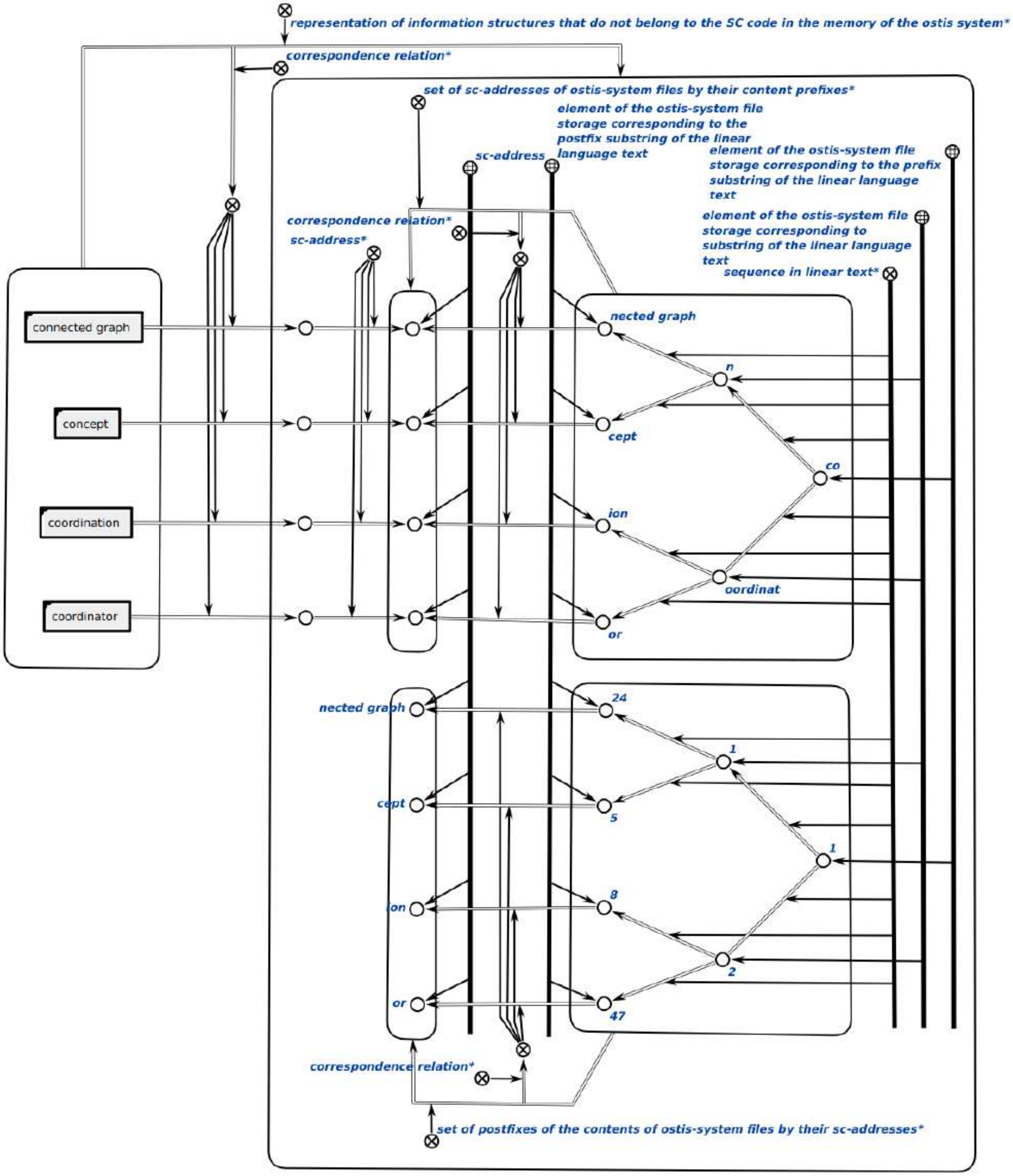

representation of information structures that do not belong to the SC code in the memory of the ostis system*
correspondence relation*
set of sc-addresses of ostis-system files by their content prefixes*
element of the ostis-system file storage corresponding to the postfix substring of the linear language text
sc-address
element of the ostis-system file storage corresponding to the prefix substring of the linear language text
element of the ostis-system file storage corresponding to substring of the linear language text
correspondence relation*
sc-address*
sequence in linear text*
connected graph
concept
coordination
coordinator
nected graph
n
cept
co
ion
oordinat
or
nected graph
cept
ion
or
24
1
5
1
8
2
47
correspondence relation*
set of postfixes of the contents of ostis-system files by their sc-addresses*

Figure 2. An example of a specification for the representation of information structures that do not belong to the SC-code in the memory of the ostis-system

- Information constructions that do not belong to the SC-code are still completely stored in RAM of the computer device on which the platform is deployed. This problem can be solved if only the first characters of substrings of information structures are stored in RAM, and the remaining parts of these substrings are stored at the file system level.
- At the moment, the information retrieval subsystem is not fully implemented. *Implementation of the ostis-system file storage* allows quickly solving the problem of searching for external information constructions by their prefix substrings, but does not allow quickly solving the problem of searching for information constructions by any substring, even for which some sample-template is specified.

The described problems will be solved within a future version of the *Software version of the ostis-platform*.

IX. General description of methods for Implementation of the sc-memory

The SCin-code and the SCfin-code are sufficient to represent the texts of the SC-code within the memory of the ostis-system. To translate some SC-code text into the ostis-system memory, it is necessary to use sc-memory methods (programs, procedures), which are elements of *Implementation of the sc-memory*.

sc-memory method

⊂ *method*

⊂ *Implementation of the sc-memory*

∋ *Method of creating an sc-storage element corresponding to the sc-node with a given type*

∋ *Method of creating an sc-storage element corresponding to the sc-arc with a given type*

∋ *Method of creating an sc-storage element corresponding to the ostis-system file with a given type*

∋ *Method of setting the information construction of the linear language in accordance with the given sc-storage element corresponding to the ostis-system file*

So, using the *Method of creating an sc-storage element corresponding to the sc-node with a given type*, the *Method of creating an sc-storage element corresponding to the sc-arc with a given type*, and the *Method of creating an sc-storage element corresponding to the ostis-system file with a given type*, it is possible to create all program elements of the *SCin-code alphabet*^ corresponding to sc-elements of the *SC-code alphabet*^, and using the *Method of setting the information construction of the linear language in accordance with the given sc-storage element corresponding to the ostis-system file* to indicate the connections between the sc-storage elements corresponding to the files of the ostis-system and external information structures represented in the ostis-storage file systems as linear text.

There are other methods in *Implementation of the sc-memory*, but they will not be covered in this article.

X. Implementation of the subsystem of network interaction with Implementation of sc-memory

A. Selected solution and its rationale

The interaction of the sc-memory software model with external resources can be carried out through a specialized programming interface (API), however, this option is inconvenient in most cases, since:

- it is only supported for a very limited set of programming languages (C, C++);
- it requires that the client application accessing the sc-memory software model actually forms a single whole with it, thus eliminating the possibility of building a distributed collective of ostis-systems;
- as a consequence of the previous paragraph, the possibility of parallel work with sc-memory of several client applications is excluded.

In order to provide the possibility of remote access to sc-memory without taking into account the programming languages with which a particular client application is implemented, it was decided to implement the possibility of accessing sc-memory using a universal language that does not depend on the means of implementing one or another component or system.

Among the effective protocols used in the implementation of client-server systems, it is worth noting the application layer protocols of the TCP/IP stack – HTTP and WebSocket protocols [59], [60]. It is advisable to use the WebSocket protocol due to the following reasons:

- WebSocket is useful in web-based systems where data sent by the server is represented or stored on the client side. In WebSocket, data is constantly transferred over the same open connection, so WebSocket communication is faster than HTTP communication [61], [62]. This is very important in terms of designing the OSTIS Ecosystem, which can consist of tens of thousands of different ostis-systems kinds.
- Since ostis-systems are based on the idea of agent-oriented knowledge processing (asynchronous processing) and the memory of such systems must be both distributed and shared, it is necessary that each of them (in particular, an independent ostis-system) be able to communicate with other ostis-systems. Moreover, such communication can and should take place on the conditions of initiating events in the memory of these systems. This implies an unambiguous conclusion that the HTTP protocol cannot be used in advanced next-generation intelligent systems

due to the unidirectional nature of the connection it creates.

A string language based on the JSON language [63], [64] – *SC-JSON-code* – was developed as a system communication language. Such choice is explained by the flexibility of setting relation between the objects it describes.

B. Implementation of the subsystem for interaction with sc-memory based on the JSON language

Generally speaking, the subsystem of interaction with the external environment can be implemented in different ways. So, for example, before the implementation of the current subsystem, there was previously its analogue in the Python programming language, which used the HTTP protocol and a binary representation of commands and responses. Therefore, there can be a wide variety of such subsystems, that can build various *Implementations of the subsystem of interaction with the external environment using network languages*.

This *Implementation of the sc-memory interaction subsystem based on the JSON language* allows ostis-systems to interact with systems from the external environment based on the generally accepted JSON data transfer transport format and provides an API for accessing the sc-memory of the sc-model interpretation platform.

Implementation of the subsystem of interaction with the external environment using network languages
⇒ *software system decomposition**:
{• *Implementation of the subsystem of interaction with the external environment using network languages based on the JSON language*
}

Implementation of the sc-memory interaction subsystem based on the JSON language
:= [Subsystem for interaction with sc-memory based on the JSON format]
∈ *non-atomic reusable ostis-systems component*
∈ *dependent reusable ostis-systems component*
∈ *client-server system*
⇒ *used method representation language**:
- *C*
- *C++*
- *Python*
- *TypeScript*
- *C#*
- *Java*

⇒ *used language**:
- *SC-JSON-code*

⇒ *software system decomposition**:
{
- *Websocket- and JSON-based server system providing network access to sc-memory*

{}
= {• *Implementation of the client system in the Python programming language*
• *Implementation of the client system in the TypeScript programming language*
• *Implementation of the client system in the C# programming language*
• *Implementation of the client system in the Java programming language*
}
}

Interaction with sc-memory is provided by transferring information in the ***SC-JSON-code*** and is conducted, on the one hand, between the server, which is part of the ostis-system, written in the same implementation language of this ostis-system and having access to its sc-memory, and, on the other hand, a set of clients who are aware of the presence of a server within the network of their usage. Using the subsystem for interaction with sc-memory based on the JSON language, it is possible to interact with the ostis-system on the same set of possible operations as in the case if the interaction took place directly, in the same implementation language of the platform for interpreting sc-models of computer systems. In this case, the result of the work differs only in the speed of information processing.

C. Concept of the SC-JSON-code

As mentioned earlier, subsystems within the implemented software version of a specialized platform communicate using the external knowledge representation language – an SC-JSON-code. This language is string, i.e. linear, and easy to reverse, since there are a large number of facilities for processing its JSON superlanguage.

SC-JSON-code
:= [Semantic JSON-code]
:= [Semantic JavaScript Object Notation code]
:= [Language of external semantic representation of knowledge based on the JSON language]
⇒ *frequently used non-primary external identifier of an sc-element**:
[sc-json-text]
∈ *common noun*
∈ *abstract language*
∈ *linear language*
⊂ *JSON*

D. Syntax and syntactic rules of the SC-JSON-code

The *SC-JSON-code syntax* is specified by: the (1) *SC-JSON-code alphabet* and the (2) SC-JSON-code grammar. In the alphabet of the SC-JSON-code, the basic syntactic classification of its elements is distinguished.

```
Syntactic classification of
SC-JSON-code elements
⊃=
{

SC-JSON-code
⇐    subset family*:
     sc-json-sentence
        ⊂    json-list of json-pairs
        ⇐    subset family*:
             sc-json-pair*
                ⇐    Cartesian product*:
                     ⟨•    sc-json-string
                      •    sc-json-object
                     ⟩
        ⇒    subdividing*:
             {•    SC-JSON-code command
              •    SC-JSON-code command response
             }

sc-json-object
⇒    subdividing*:
     {•    sc-json-list
      •    sc-json-pair
      •    sc-json-literal
             ⇒    subdividing*:
                  {•    sc-json-string
                   •    sc-json-number
                  }
     }
}
```

The *SC-JSON-code alphabet^* is a set of all possible characters in the SC-JSON-code. Since the *SC-JSON-code* is a linear string knowledge representation language, its alphabet includes the combination of the alphabets of all languages, the texts in which can represent external identifiers and/or the contents of ostis-system files, the set of all digits, and the set of all other special characters. Alphabet sequences can form sc-json-keywords, *sc-json-pairs*, *sc-json-sentences* from *sc-json-pairs*, and *sc-json-texts* from *sc-json-sentences*. At the same time, constructions on the SC-JSON-code are built according to the following syntactic rules:

- Each *SC-JSON-code grammar* rule describes the correct order of sc-json objects in an sc-json-sentence according to the *SC-JSON-code syntax*. The set of *SC-JSON-code grammar* rules describes the order of sc-json-sentences in sc-json-text that is correct in terms of the *SC-JSON-code syntax*. Each sc-json-sentence is an sc-json-list of sc-json-pairs, which represents a command or response to that command.
- Each *command (command response) in the SC-JSON-code* consists of a header that includes sc-json-pairs describing the command itself (command response) and a message that is different for each class of commands (command responses). The *command (command response) message in the SC-JSON-code* is usually a list of sc-json-objects, which may not be limited in size.
- Each sc-json-pair consists of two elements: a keyword and some other sc-json-object associated with that keyword. The set of keywords in sc-json-pairs is determined by a specific class of *commands (command response) in the SC-JSON-code*. The sc-json pair starts with an open brace "{" and ends with a close brace "}". The keyword and the sc-json object associated with it are separated by a colon character ":".
- Sc-json strings written in sc-json texts begin and end with the double-quoted character “.
- Sc-json-lists that do not consist of sc-json-pairs begin with an opening square bracket "[" and end with a close square bracket "]". Sc-json-objects in sc-json-lists are separated by commas ",".

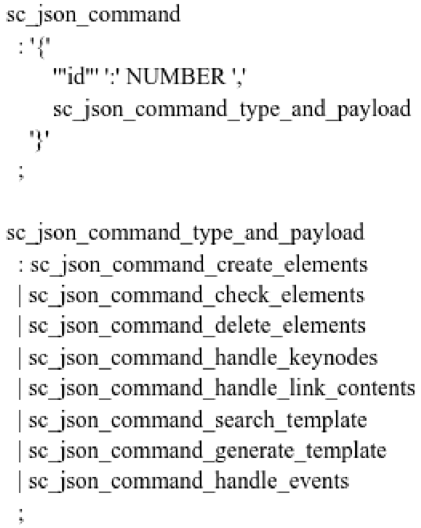
E. Syntax and grammar of the SC-JSON-code. SC-JSON command and response examples

The grammar of the SC-JSON-code is the set of all possible rules used in building commands and responses to them in the SC-JSON code. Each *SC-JSON-code* command has a unique *SC-JSON-code* grammar rule. The *SC-JSON grammar* rules allow correctly representing commands in the SC-JSON-code. Each *SC-JSON-code* grammar rule is represented as a rule in the *ANTLR Grammar Description Language* and its natural language interpretation.

```
SC-JSON grammar
∋    key sc-element′:
     Rule that specifies the syntax of SC-JSON-code
     commands
        ⇐    syntax rule*:
             SC-JSON-code command
∋    key sc-element′:
     Rule that specifies the syntax of SC-JSON-code
     command responses
        ⇐    syntax rule*:
             SC-JSON-code command response
∋    Rule that specifies the syntax of the command for
     creating sc-elements
        ⇐    syntax rule*:
             command for creating sc-elements
```

∋ *Rule that specifies the syntax of response to the command for creating sc-elements*
  ⇐ *syntax rule**:
    response to the command for creating sc-elements

The rule that specifies the syntax of the *SC-JSON-code command* means the following 3. The *SC-JSON-code command* class includes the *command for creating sc-elements*, *command for getting corresponding types of sc-elements*, *command for deleting sc-elements*, *command for processing key sc-elements*, *command for processing contents of ostis-system files*, *command for searching for sc-constructions isomorphic to a given sc-template*, *command for generating an sc-constructions isomorphic to a given sc-template*, and *command processing sc-event*. The *SC-JSON-code command* includes the command ID, type, and message.

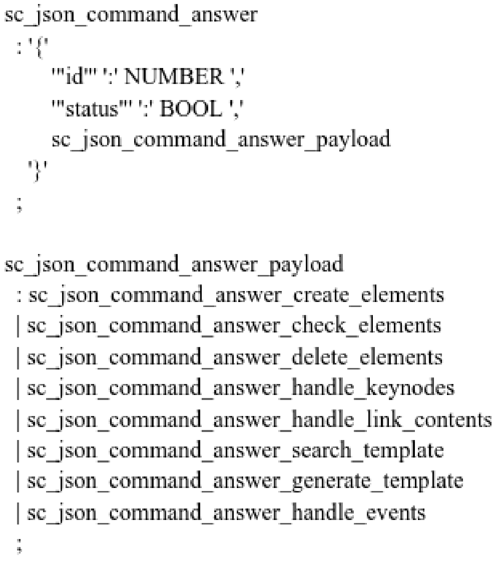


```
sc_json_command
  : '{'
      '"id"' ':' NUMBER ','
      sc_json_command_type_and_payload
    '}'
  ;

sc_json_command_type_and_payload
  : sc_json_command_create_elements
  | sc_json_command_check_elements
  | sc_json_command_delete_elements
  | sc_json_command_handle_keynodes
  | sc_json_command_handle_link_contents
  | sc_json_command_search_template
  | sc_json_command_generate_template
  | sc_json_command_handle_events
  ;
```

Figure 3. Description of the Rule that specifies the syntax of the *SC-JSON-code command*

The rule specifying the syntax of the *SC-JSON-code command response* describes the syntax of command responses described by the previous rule. The *SC-JSON-code command response* class includes the *command response for creating sc-elements*, *command response for getting the corresponding types of sc-elements*, *command response for deleting sc-elements*, *command response for processing key sc-elements*, *command response for processing contents of ostis-system files*, *command response for searching for sc-constructions isomorphic to the given sc-template*, *command response for generating an sc-construction isomorphic to the given sc-template*, and *command response for sc-event processing*.



```
sc_json_command_answer
  : '{'
      '"id"' ':' NUMBER ','
      '"status"' ':' BOOL ','
      sc_json_command_answer_payload
    '}'
  ;

sc_json_command_answer_payload
  : sc_json_command_answer_create_elements
  | sc_json_command_answer_check_elements
  | sc_json_command_answer_delete_elements
  | sc_json_command_answer_handle_keynodes
  | sc_json_command_answer_handle_link_contents
  | sc_json_command_answer_search_template
  | sc_json_command_answer_generate_template
  | sc_json_command_answer_handle_events
  ;
```

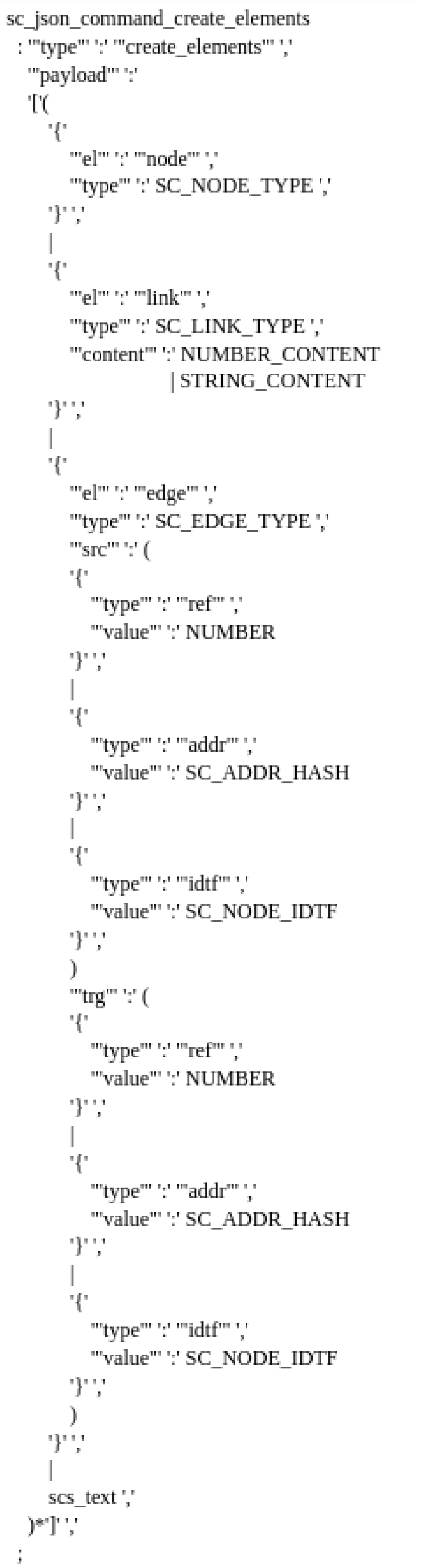
Figure 4. Description of the Rule that specifies the syntax of the *SC-JSON-code command response*

The *command for creating sc-elements* message contains a list of descriptions of the sc-elements to be created. Such sc-elements can be an sc-node, an sc-arc, an sc-edge, or an ostis-system file. The sc-element type is specified in pair with the "el" keyword: for an sc-node, the sc-json-type of element is represented as a "node", for an sc-arc and an sc-edge – an "edge", for ostis-system file – a "link". Type labels of sc-elements are specified in their corresponding descriptions in the command message, paired with the "type" keyword. If the sc-element being created is an ostis-system file, then the contents of this ostis-system file are additionally specified in pair with the "content" keyword; if the sc-element being created is an sc-arc or an sc-edge, then the descriptions of the sc-elements they go out and the sc-elements they come in are specified. Descriptions of such sc-elements consist of two pairs: the first pair indicates the method of association with the sc-element and is represented as "addr", or "idtf", or "ref" paired with the "type" keyword, the second pair represents what is associated with this sc-element: its hash, system identifier, or number in the array of created sc-elements – paired with the "value" 5 keyword.

The *command response for creating sc-elements* message is a list of hashes of created sc-elements corresponding to the *command for creating sc-elements* descriptions with status 1, in case of successful processing of the 6 command.

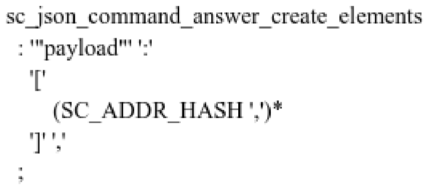
The *SC-JSON-code command* set is easily extensible due to the flexibility and functionality of the JSON language. The set of the *command responses in the SC-JSON-code* is also easily extensible, along with the *SC-JSON-code commands* extension.

command for creating sc-elements
:= [create elements command]
⊂ *SC-JSON-code command*



```
sc_json_command_create_elements
  : '"type"' ':' '"create_elements"' ','
    '"payload"' ':'
    '['(
       '{'
          '"el"' ':' '"node"' ','
          '"type"' ':' SC_NODE_TYPE ','
       '}' ','
       |
       '{'
          '"el"' ':' '"link"' ','
          '"type"' ':' SC_LINK_TYPE ','
          '"content"' ':' NUMBER_CONTENT
                       | STRING_CONTENT
       '}' ','
       |
       '{'
          '"el"' ':' '"edge"' ','
          '"type"' ':' SC_EDGE_TYPE ','
          '"src"' ':' (
          '{'
             '"type"' ':' '"ref"' ','
             '"value"' ':' NUMBER
          '}' ','
          |
          '{'
             '"type"' ':' '"addr"' ','
             '"value"' ':' SC_ADDR_HASH
          '}' ','
          |
          '{'
             '"type"' ':' '"idtf"' ','
             '"value"' ':' SC_NODE_IDTF
          '}' ','
          )
          '"trg"' ':' (
          '{'
             '"type"' ':' '"ref"' ','
             '"value"' ':' NUMBER
          '}' ','
          |
          '{'
             '"type"' ':' '"addr"' ','
             '"value"' ':' SC_ADDR_HASH
          '}' ','
          |
          '{'
             '"type"' ':' '"idtf"' ','
             '"value"' ':' SC_NODE_IDTF
          '}' ','
          )
       '}' ','
       |
       scs_text ','
    )*']' ','
  ;
```

Figure 5. Description of the Rule that specifies the syntax of the *command for creating sc-elements*



```
sc_json_command_answer_create_elements
  : '"payload"' ':'
    '['
       (SC_ADDR_HASH ',')*
    ']' ','
  ;
```

Figure 6. Description of the Rule that specifies the syntax of the *command response for creating sc-elements*

⇒ *example**:
Example of the command for creating sc-elements
⇒ *command class**:
command response for creating sc-elements

The Websocket- and JSON-based server system providing network access to sc-memory will interpret the *Example of command for creating sc-elements* 7 as "Process command for creating sc-elements: an sc-node of type 1 (of an unspecified type), an ostis-system file of type 2 (of an unspecified type), and contents in the form of a floating point number 45.4, and an sc-arc of type 32 (of a constant type) between the sc-element located at the zero position in the array of created sc-elements, and an sc-element in the first position in the same array".

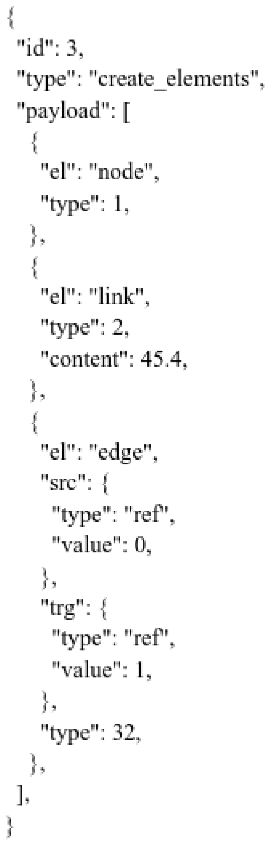
It should be noted that at the sc-memory interface level, the command is interpreted quickly due to the fact that templates for creating constructions isomorphic to them are not used. Also, the contents of the message of the *command for creating sc-elements* can be empty.

command response for creating sc-elements
:= [create elements command response]
⊂ *SC-JSON-code command response*
⇒ *example**:
Example of the command response for creating sc-elements

An example of the command response for creating sc-elements is an example of a response to the previous command if this command was interpreted and executed successfully 8.

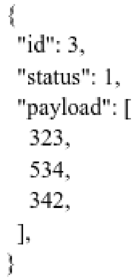
The formal text of the *Example of the command response for create sc-elements* is equivalent to the natural language text "Created sc-elements with hashes 323, 534, and 342, respectively. The command was processed successfully".

A detailed description of the syntax of commands and responses to these commands, as well as their examples, can be found in the OSTIS Standard [6].



```
{
 "id": 3,
 "type": "create_elements",
 "payload": [
  {
   "el": "node",
   "type": 1,
  },
  {
   "el": "link",
   "type": 2,
   "content": 45.4,
  },
  {
   "el": "edge",
   "src": {
    "type": "ref",
    "value": 0,
   },
   "trg": {
    "type": "ref",
    "value": 1,
   },
   "type": 32,
  },
 ],
}
```

Figure 7. An example of the *command for creating sc-elements*



```
{
 "id": 3,
 "status": 1,
 "payload": [
  323,
  534,
  342,
 ],
}
```

Figure 8. An example of the *command response for creating sc-elements*

F. Description of Implementation of The server system based on Websocket and JSON, providing network access to sc-memory

The *Server system based on Websocket and JSON, providing network access to sc-memory* is an interpreter of commands and responses of the *SC-JSON-code* for programm representation of sc-constructions in sc-memory using the *Library of software components for processing json texts (JSON for Modern C++)* and the *Library of cross-platform software components for implementing server applications based on Websocket (WebSocket++)*, and is also provided with comprehensive test coverage through the Google Tests and Google Benchmark Tests software frameworks. The *Library of software components for processing json-texts (JSON for Modern C++)* has a rich, convenient, and high-speed functionality necessary for the implementation of such components of ostis-systems, and the *Library of cross-platform software components for the implementation of server applications based on Websocket (WebSocket++)* allows elegantly designing server systems without using redundant dependencies and solutions. The software component is configured with the help of the *Software component for ostis-systems software components configuration*, as well as CMake and Bash scripts.

Implementation of the Server system based on Websocket and JSON, providing network access to sc-memory

:= [Implementation of the Websocket-based system that provides parallel-asynchronous multi-client access to sc-memory of the sc-model interpretation platform using the SC-JSON-code]

:= [sc-json-server]

⇒ *frequently used non-primary external identifier of the sc-element**:
[sc-server]

∈ *atomic reusable ostis-systems component*

∈ *dependent reusable ostis-systems component*

⇒ *used method representation language**:
- *C*
- *C++*

⇒ *used language**:
- *SC-JSON-code*

⇒ *component dependencies**:
{• *Library of software components for processing json-texts JSON for Modern C++*
- *Library of cross-platform software components for implementing server applications based on Websocket WebSocket++*
- *Software component for ostis-systems software components configuration*
- *Implementation of the sc-memory*

}

It is worth noting that the current *Implementation of the Server system based on Websocket and JSON, providing network access to sc-memory* is not the first of its kind and replaces its previous implementation written in Python. The reason for this replacement is as follows:

- Previous *Implementation of the server system based on Websocket and JSON, providing access to sc-memory using SC-JSON-code commands*, implemented in the Python programming language, depends on the Boost Python library provided by

the C++ Language Development and Collaboration Community, as well as Python. The fact is that such a solution requires the support of the mechanism for interpreting the Python program code into the C++ language, which is redundant and unreasonable, since most of the *Software implementation of the ostis-platform* program code is implemented in the C and C++ languages. The new implementation of the described software system allows getting rid of the usage of capacious and resource-intensive libraries (for example, boost-python-lib, llvm) and the Python language.
- When implementing distributed subsystems, the speed of knowledge processing plays an important role, that is, the ability to quickly and urgently respond to user requests. The quality of access to sc-memory through the implemented *Subsystem for interacting with sc-memory based on the JSON language* should be commensurate with the quality of access to sc-memory using a specialized API, implemented in the same programming language as the system itself. The new implementation makes it possible to increase the processing speed of requests by the *JSON-based sc-memory interaction subsystem*, including knowledge processing, by at least 1.5 times compared to the previous implementation of this subsystem.

Implementation of the Server system based on Websocket and JSON, providing network access to sc-memory possesses the following common properties:

- From the point of view of its model, the server subsystem has the same specialized programming interface as the *Implementation of the sc-memory*, however, interaction with it using such an interface is carried out via the network. This makes it possible for client systems implemented in different programming languages to interact with the same shared memory.
- This subsystem can be considered as an interpreter of an external knowledge representation language (SC-JSON-code), which can be used by ostis-systems implemented on the basis of a specialized ostis-platform. Each command and response to a command of this language corresponds to a handler (potentially an agent at all), which is part of this interpreter. The SC-JSON-code language of external knowledge representation itself is independent of the platform implementation and is used only as a language of external knowledge representation, however, it can be used when implementing other tools and interpreters of sc-models of ostis-systems.
- The implemented software component provides multi-user asynchronous access to sc-memory. While testing the sc-server, it turned out that its implementation allows processing requests from at least 1000 client systems. Due to the need to provide parallel access to sc-memory, synchronization blocks were added at the implementation level of the software component. For example, in the implementation, it is possible to notice a queue of commands for processing by the system – regardless of the number of client systems and how many commands were sent for processing, all commands can queue up. This solution allows temporarily bypassing the problems of interaction of synchronization blocks at the sc-memory level when processing different types of commands over it (search, generative, destructive, etc.). However, the server system cannot be shut down as long as the command queue has any pending commands. Also, the server system continues to work if the list of client system identifiers still has non-disconnected ones. The need for these functions of the server subsystem is determined by the need to support the atomicity of requests processed by the system.
- In the process of testing the subsystem, an estimate of its speed of processing commands and responses was obtained. During load testing, a test client system was used, written in C++ and not possessing the functionality of processing texts of the SC-JSON-code. As a result of testing, it was found that when sending 1000 different commands – commands for creating sc-elements, commands for processing the contents of ostis-system files, and commands for deleting sc-elements – the time spent on their processing did not exceed 0.2 seconds. At the same time, in some cases, processing 1000 commands for creating sc-elements took no more than 0.14 seconds, commands for deleting sc-elements – no more than 0.07 seconds, commands for processing the contents of ostis-system files – no more than 0.27 seconds, commands search for sc-constructions isomorphic to a given sc-template – no more than 0.45 seconds.

The *Server system based on Websocket and JSON, providing network access to sc-memory* describes the necessary and sufficient programming interface for interacting with sc-memory. In the general case, it describes the functionality of not only the *Server system based on Websocket and JSON, providing network access to the sc-memory* but also the client systems interacting with it, since these client systems often include a specialized programming interface similar to the server system interface but implemented in a different programming language.

XI. Implementation of the interpreter of user interface sc-models

A. Concept of the interpreter of user interface sc-models

In most cases, user interface development in modern systems takes up most of the time spent on developing the entire system. However, the effectiveness of using a software system depends on the developed user interface [65].

Along with the *Implementation of the sc-memory*, an important part of the *Software implementation of the ostis-platform* is the *Implementation of the interpreter of user interface sc-models*, which provides basic tools for viewing and editing the knowledge base by the user, tools for navigation through the knowledge base (asking questions to the knowledge base) and can be supplemented with new components depending on the problems solved by each specific ostis-system.

Implementation of the interpreter of user interface sc-models
∈ *non-atomic reusable ostis-systems component*
∈ *dependent reusable ostis-systems component*
⇒ *used method presentation language**:
- *JavaScript*
- *TypeScript*
- *Python*
- *HTML*
- *CSS*

⇒ *component dependencies**:
{• *Library of standard interface components in the JavaScript programming language*
• *Library for implementing server applications in the Python programming language, named Tornado*
• *Implementation of the client system in the TypeScript programming language*
• *Implementation of the client system in the Python programming language*
}

An important principle of the *Implementation of the interpreter of user interface sc-models* is the simplicity and uniformity of connecting any user interface components (editors, visualizers, switches, menu commands, etc.). To do this, the Sandbox software layer is implemented, within which low-level operations of interaction with the server part are implemented and which provides a more convenient programming interface for component developers. The current version of the *Implementation of the interpreter of user interface sc-models* is open and available at [66].

B. Main components of the interpreter of user interface sc-models

Implementation of the interpreter of user interface sc-models
⇒ *software system decomposition**:
{• *User interface command menu bar*
• *Component for switching the language of identification of displayed sc-elements*
• *Component for switching the external language of knowledge visualization*
• *Search field of sc-elements by identifier*
• *Panel for displaying the user dialog with the ostis-system*
• *Knowledge visualization and editing panel*
⇒ *software system decomposition**:
{• *Visualizer of sc.n-texts*
• *Visualizer and editor of sc.g-texts*
}
}

The *Component for switching the language of identification of displayed sc-elements* is an image of the set of natural languages available in the system. User interaction with this component switches the user interface to a mode of communication with a specific user with the help of *basic sc-identifiers* belonging to this *natural language* (Fig. 9). This means that when displaying sc-identifiers of sc-elements in any language, for example, SCg-code or SCn-code, *basic sc-identifiers* belonging to the given *natural language* will be used. This applies both to sc-elements displayed within the *Knowledge visualization and editing panel* and any other sc-elements, for example, command classes and even *natural languages* themselves, displayed within the *Component for switching the language of identification of displayed sc-elements of the ostis-meta-system*.

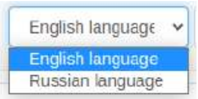


Figure 9. The Component for switching the language of identification of displayed sc-elements of the ostis-meta-system

The *Component for switching the external language of knowledge visualization* is used to switch the knowledge visualization language in the current window displayed on the *Knowledge visualization and editing panel*. In the current implementation, SCg-code and SCn-code (Fig. 10, as well as any other languages included in the *external SC-code visualization languages* set, are supported by default as such languages.

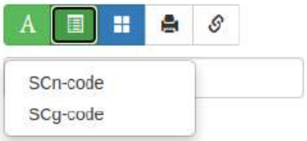
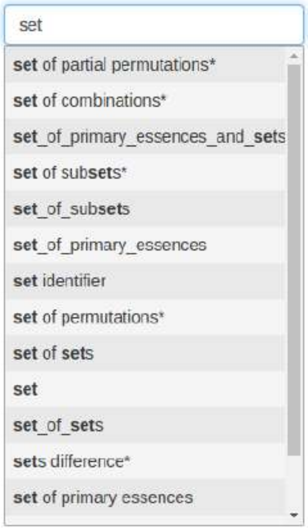


Figure 10. The Component for switching the external language of knowledge visualization of the ostis-meta-system

The *Search field for sc-elements by identifier* allows searching for sc-identifiers containing the substring entered in this field (case sensitive). As a result of the search, a list of sc-identifiers containing the specified substring is displayed (Fig. 11), when interacting with them, the question “What is this?” is automatically set, the argument of which is either for the sc-element itself, which has the given sc-identifier (if the specified sc-identifier is the main or system identifier, and thus the specified sc-element can be uniquely determined), or for the internal file of the ostis-system that is the sc-identifier (in case when the given sc-identifier is not the main one).



set

set of partial permutations*

set of combinations*

set_of_primary_essences_and_**sets**

set of sub**sets***

set_of_sub**sets**

set_of_primary_essences

set identifier

set of permutations*

set of **sets**

set

set_of_**sets**

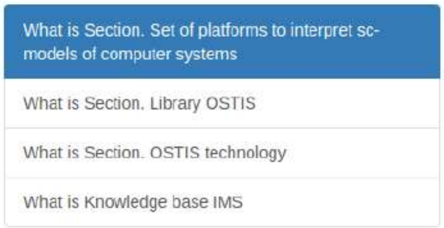
sets difference*

set of primary essences

Figure 11. The Component for switching the external language of knowledge visualization of the ostis-meta-system

The *Panel for displaying the user dialog with the ostis-system* displays a time-ordered list of sc-elements (Fig. 12) that are signs of actions initiated by the user within the dialog with the ostis-system by interacting with images of the corresponding command classes (that is, if the action was initiated in another way, for example, by explicitly initiating it by creating an arc of membership to the *action initiated* set in the sc.g editor, then it will not be displayed on this panel). When the user interacts with any of the depicted action signs, the *Knowledge visualization and editing panel* displays a window containing the result of this *action* in the visualization language, in which it was displayed when the user viewed it in the last (previous) once. Thus, in the current implementation, this panel can work only if the action initiated by the user assumes the result of this action explicitly represented in memory. In turn, it follows from this that at present this panel, as well as the whole *Implementation of the interpreter of user interface sc-models*, allows working with the system only in the “question-answer” dialog mode.

The *Knowledge visualization and editing panel* displays windows containing sc-text, represented in some language from the set of *external SC-code visualization languages* and, as a rule, the result of some action initiated by the user. If the corresponding visualizer supports the ability to edit texts of the corresponding natural language, then it is also an editor at the same time. If necessary, the user interface of each specific ostis-system can be supplemented with visualizers and editors of various external languages, which in the current version of *Implementation of the interpreter of user interface sc-models* will also be located on the *Knowledge visualization and editing panel*. By default, two visualization and editing panels are available: the *Visualizer of sc.n-texts* (Fig. 13) and the *Visualizer and editor of sc.g-texts* (Fig. 14).



What is Section. Set of platforms to interpret sc-models of computer systems

What is Section. Library OSTIS

What is Section. OSTIS technology

What is Knowledge base IMS

Figure 12. The Panel for displaying the user dialog with the ostis-meta-system

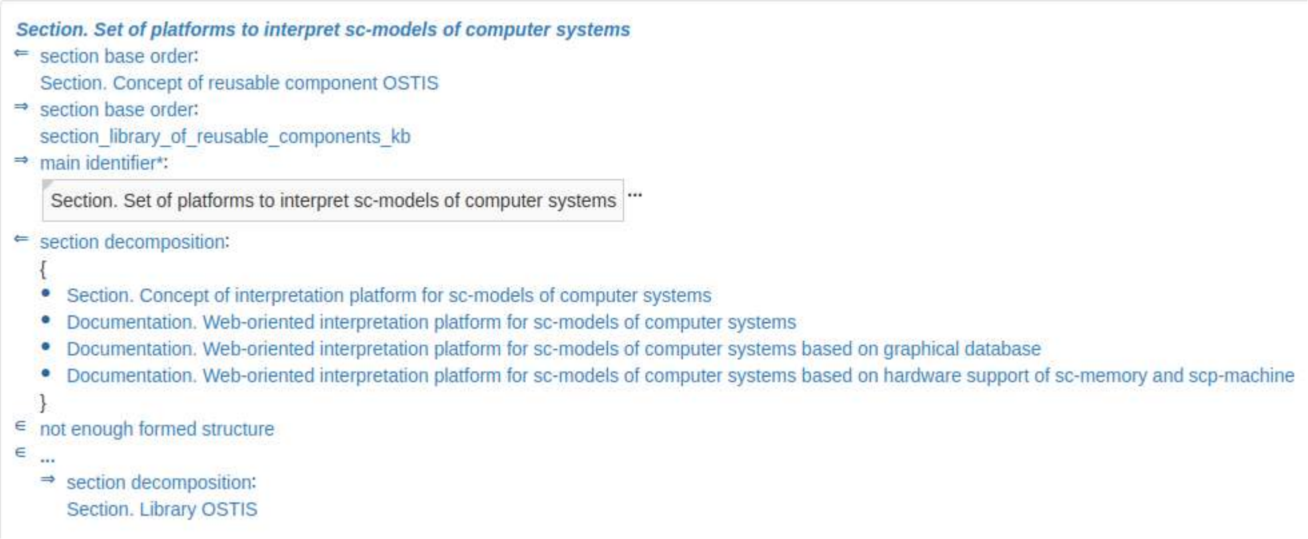
The *User interface commands menu bar* contains images of command classes (both atomic and non-atomic) currently available in the knowledge base and included in the *main user interface* decomposition (meaning the complete decomposition, which in may include several levels of non-atomic instruction classes in general) (Fig. 15). Interaction with the image of a non-atomic instruction class initiates a command for the image of instruction classes included in the decomposition of this non-atomic instruction class. Interaction with the image of an atomic command class initiates the generation of a command of this class with previously selected arguments based on the corresponding *generalized command class statement* (command class template).

The semantic models of the described user interface components are represented in more detail in [67].

C. Advantages and disadvantages represented in the current version of Implementation of the interpreter of user interface sc-models

The current implementation of the sc-interface model interpreter has a large number of shortcomings, namely:

- The idea of platform independence of the user interface (building the sc-model of the user interface) is not fully implemented. Fully describing the sc-model of the user interface (including the exact placement, size, design of components, their behavior, etc.) is currently likely to be difficult due to performance limitations, but it is quite possible to implement the ability to ask questions to all interface components, change them location, etc., however, these features cannot be implemented in the current version of the platform implementation.



Section. Set of platforms to interpret sc-models of computer systems

⇐ section base order:
Section. Concept of reusable component OSTIS

⇒ section base order:
section_library_of_reusable_components_kb

⇒ main identifier*:
Section. Set of platforms to interpret sc-models of computer systems ...

⇐ section decomposition:
{
- Section. Concept of interpretation platform for sc-models of computer systems
- Documentation. Web-oriented interpretation platform for sc-models of computer systems
- Documentation. Web-oriented interpretation platform for sc-models of computer systems based on graphical database
- Documentation. Web-oriented interpretation platform for sc-models of computer systems based on hardware support of sc-memory and scp-machine

}

∈ not enough formed structure

∈ ...
⇒ section decomposition:
Section. Library OSTIS

Figure 13. The Visualizer of sc.n-texts of the ostis-meta-system

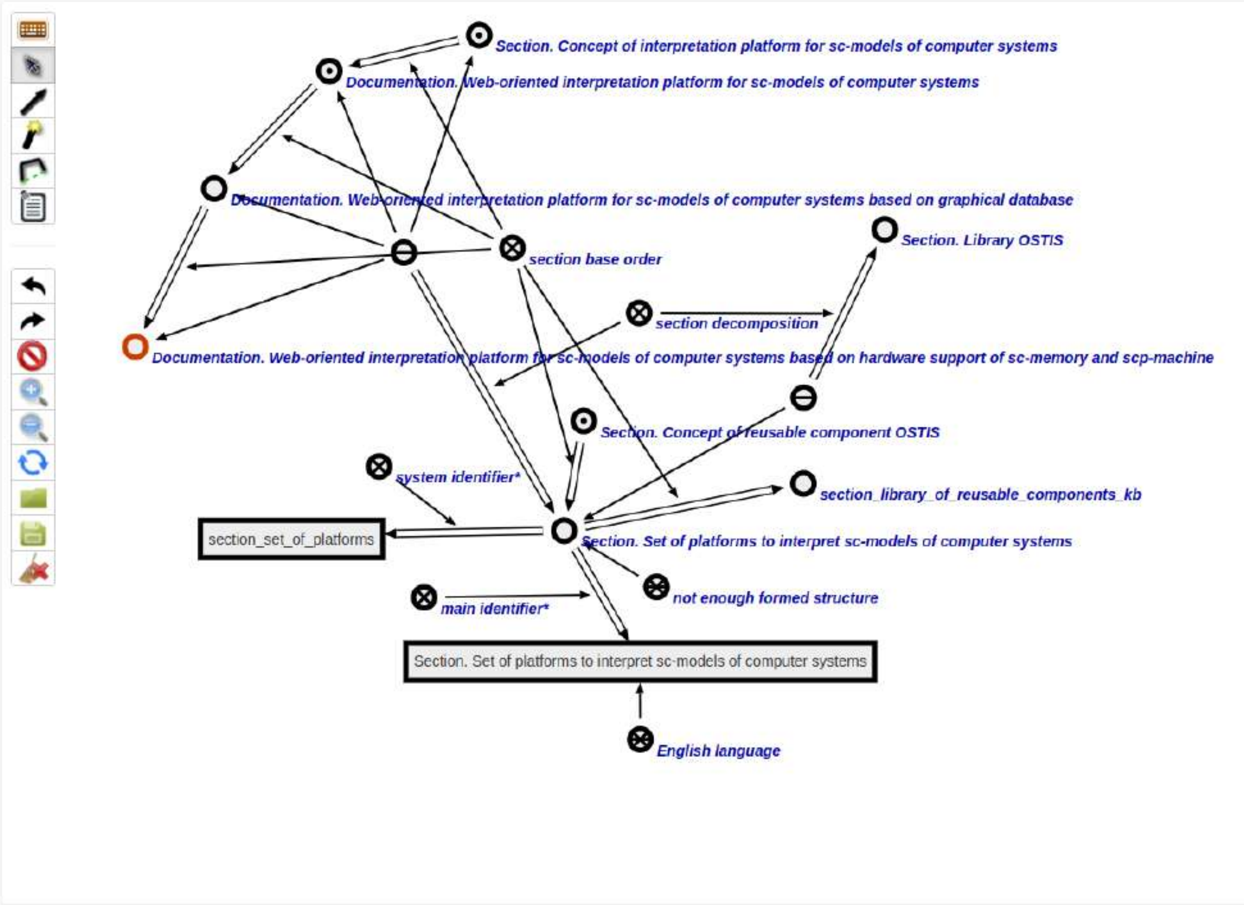

Section. Concept of interpretation platform for sc-models of computer systems
Documentation. Web-oriented interpretation platform for sc-models of computer systems
Documentation. Web-oriented interpretation platform for sc-models of computer systems based on graphical database
Section. Library OSTIS
section base order
section decomposition
Documentation. Web-oriented interpretation platform for sc-models of computer systems based on hardware support of sc-memory and scp-machine
Section. Concept of reusable component OSTIS
system identifier*
section_library_of_reusable_components_kb
section_set_of_platforms
Section. Set of platforms to interpret sc-models of computer systems
not enough formed structure
main identifier*
Section. Set of platforms to interpret sc-models of computer systems
English language

Figure 14. The Visualizer and editor of sc.g-texts of the ostis-meta-system

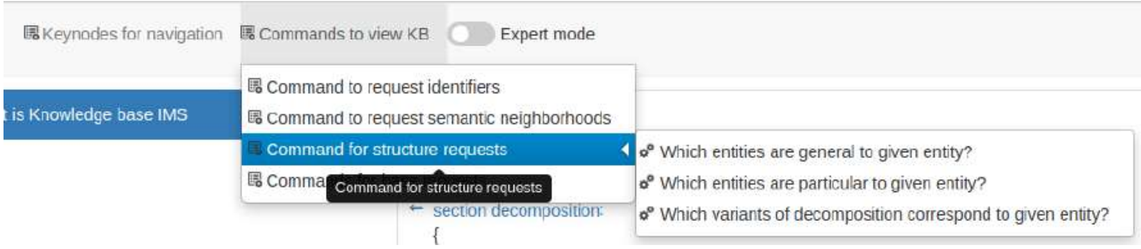

Keynodes for navigation
Commands to view KB
Expert mode
is Knowledge base IMS
Command to request identifiers
Command to request semantic neighborhoods
Command for structure requests
Comma
Command for structure requests
section decomposition:
{
Which entities are general to given entity?
Which entities are particular to given entity?
Which variants of decomposition correspond to given entity?

Figure 15. The User interface commands menu bar of the ostis-meta-system

- In addition, part of the interface actually works directly with sc-memory using WebSocket technology and part through an interlayer based on the tornado library for the Python programming language, which leads to additional dependencies on third-party libraries. Recently, the development of the current *Software implementation of the ostis-platform* has largely solved this problem, but there are still components implemented in Python.
- Some of the components (for example, the search field by identifier) are implemented by third-party tools and have almost nothing to do with sc-memory. This hinders the development of the platform.
- The current *Implementation of the sc-model interpreter for user interfaces* is focused only on dialog with the user (in the style of a user question – a system answer). Obviously, necessary situations are not supported, such as executing a command that does not expect a response; the occurrence of an error or lack of response; the need for the system to ask a question to the user, etc.
- Restricted user interaction with the system without the usage of special controls. For example, it is possible to ask the system a question by drawing it in the SCg-code, but the user will not see the answer, although it will be generated in memory by the corresponding agent. Most of the technologies used in the implementation of the platform are now outdated, which hinders the development of the platform.
- There is no inheritance mechanism implemented when adding new external languages. For example, adding a new language, even one that is very close to the SCg-code, requires physically copying the component code and making the appropriate changes, which results in two unrelated components that begin to develop independently of each other.
- Low level of documenting the current *Implementation of the interpreter of user interfaces sc-models*. Represented current specification only describes the key points of the *Implementation of user interfaces sc-models* but does not cover them.

Based on the shortcomings described, the following requirements are imposed on future implementation:

- Unify the principles of interaction of all interface components with the *Implementation of the sc-memory*, regardless of what type the component belongs to. For example, a list of menu commands should be formed through the same mechanism as a response to a user request, an editing command generated by the user, a command for adding a new fragment to the knowledge base, etc. It is necessary to improve the ways of using the interface for convenient and comfortable usage [68].
- Unify the principles of user interaction with the system, regardless of the mode of interaction and the external language. For example, it should be possible to ask questions and execute other commands directly through the SCg/SCn interface. At the same time, it is necessary to take into account the principles of editing the knowledge base so that the user cannot, under the guise of asking a question, enter new information into the agreed part of the knowledge base.
- Unify the principles of handling events that occur during user interaction with interface components – the behavior of buttons and other interactive components should not be set statically by third-party tools but implemented as an agent, which, nevertheless, can be implemented in an arbitrary way (not necessarily on platform-independent level). Any action performed by the user, at the logical level, must be interpreted and processed as the initiation of the agent.
- Provide the ability to execute commands (in particular, ask questions) with an arbitrary number of arguments, including without arguments.
- Make it possible to display the answer to a question in parts if the answer is very large and takes a long time to display.
- Each displayed interface component should be interpreted as an image of some sc-node described in the knowledge base. Thus, the user should be able to ask arbitrary questions to any interface components.
- Simplify and document the mechanism for adding new components as much as possible.
- Provide the ability to add new components based on existing ones without creating independent copies.

For example, it should be possible to create a component for a language that extends the SCg language with new primitives, redefine the principles for placing sc-texts, etc.

- Minimize dependency on third-party libraries.
- Minimize the usage of the HTTP protocol (bootstrap of the common interface structure), ensure the possibility of equal two-way interaction between the server and client parts.

Obviously, the implementation of most of the above requirements is associated not only with the implementation of the platform itself but also requires the development of the theory of logical-semantic models of user interfaces and the refinement of the general principles for organizing user interfaces of ostis-systems within it. However, the fundamental possibility of implementing such models should be taken into account in the *Implementation of the ostis-platform*.

XII. Plans for the development of the *Software implementation of the ostis-platform*

In the further development of the *Software implementation of the ostis-platform*, it will be important and correct to:

- maximally detail the specification of the components of the designed ostis-platform, including the languages used for external and internal knowledge representation, and clearly stratify the hierarchy of classes and relations used in describing the components of the ostis-platforms;
- eliminate and take into account the shortcomings in the implementation of new components in the designed ostis-platform, indicate possible options for their implementation;
- reduce the dependency of the ostis-platform components to a minimum, that is, if possible, implement them in the SCP language (for example, an interpreter of user interface sc-models);
- evaluate the quality of the designed system and its components as a whole.

In the direction of improving the quality and efficiency of the *Software implementation of the ostis-platform* components, the following problems will be solved first:

- The implemented sc-memory model is not intended for its usage in a multi-user mode, especially when there are more than 4 subjects of interaction with it. This, in turn, hinders the implementation of all the principles of the OSTIS Technology. All this is explained by the failure of the implementation of blocking mechanisms at the level of the memory itself. The sc-memory model will be revised and improved in such a way as to reduce the usage of blocking mechanisms and minimize the number of mutually exclusive situations for processes in sc-memory.
- Using the current *Software implementation of the ostis-platform* is quite difficult and resource-intensive. This is primarily due to the lack of the possibility for collectively developing knowledge bases. This is affected by the lack of the necessary interface components for easy editing and viewing the knowledge base. For example, the current scg-editor is quite primitive and inconvenient to use, and the tools for creating methods (programs) are not implemented at all.

XIII. Conclusion

The current implementation of the ostis-platform is a universal tool for designing next-generation computer systems. It acts as a software emulator of a semantic associative computer (!), focused on the semantic representation and processing of information of any kind. The ostis-platform acts as a program memory for any next-generation software c.s., implemented according to the principles of the OSTIS Technology, in which a logical-semantic model (knowledge) can be placed, regardless of its type and contents. Thus, on the basis of the ostis-platform, the sc-model of the OSTIS Metasystem is implemented [69], which acts as a software implementation of the OSTIS Standard [6].

Using the ostis-platform, it is possible to solve any information problems of human activity. In this sense of the word, the implemented ostis-platform is a design automation system not only for other systems but also for solving information problems of any kind in general.

In this article, the problems of ensuring the design of platforms for the design and development of other systems are considered. A comparative analysis of existing solutions in the field of design automation of c.s. and justified the chosen decision in detail. The work determines the solution of the problem in the form of designing and developing universal interpreters of logical-semantic models of systems according to the principles underlying the OSTIS Technology, named an ostis-platform. This article is also a formal specification of the first *Software implementation of the ostis-platform.*

Acknowledgment

The author would like to thank the research groups of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics and the Brest State Technical University for their help in the work and valuable comments.

References

[1] A. Iliadis, “The tower of babel problem: making data make sense with basic formal ontology,” *Online Information Review*, vol. 43, no. 6, pp. 1021–1045, 2019.

[2] S. C. J. Lim, Y. Liu, Y. Chen *et al.*, “Ontology in design engineering: status and challenges,” 2015.

[3] I. Ahmed, G. Jeon, and F. Piccialli, "From artificial intelligence to explainable artificial intelligence in industry 4.0: a survey on what, how, and where," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5031–5042, 2022.

[4] Jeff Waters and Brenda J. Powers and Marion G. Ceruti, "Global interoperability using semantics, standards, science and technology (gis3t)," *Computer Standards & Interfaces*, vol. 31, no. 6, pp. 1158–1166, 2009.

[5] V. Golenkov, N. Guliakina, V. Golovko, and V. Krasnoproshin, "On the current state and challenges of artificial intelligence," in *International Conference on Open Semantic Technologies for Intelligent Systems*. Springer, 2022, pp. 1–18.

[6] Golenkov Vladimir and Guliakina Natalia and Shunkevich Daniil, *Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[7] Sokolov A.P., Golubev A.O., "Computer-aided design system for composite materials. part 3. graph-oriented methodology for developing user-system interaction tools," *Izvestiya SPbGETU LETI*, pp. 43–57, 2021.

[8] T. S. Dillon, E. Chang, and P. Wongthongtham, "Ontology-based software engineering-software engineering 2.0," in *19th Australian Conference on Software Engineering (ASWEC 2008)*. IEEE, 2008, pp. 13–23.

[9] Dillon, Tharam and wu, Chen and Chang, Elizabeth, "Gridspace: Semantic grid services on the web-evolution towards a softgrid," in *3rd International Conference on Semantics, Knowledge, and Grid, SKG 2007*, 11 2007, pp. 7–13.

[10] V. Kabilan, "Ontology for information systems (04is) design methodology: conceptualizing, designing and representing domain ontologies," Ph.D. dissertation, KTH, 2007.

[11] A. M. Ouksel and A. Sheth, "Semantic interoperability in global information systems," *ACM Sigmod Record*, vol. 28, no. 1, pp. 5–12, 1999.

[12] F. W. Neiva, J. M. N. David, R. Braga, and F. Campos, "Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature," *Information and Software Technology*, vol. 72, pp. 137–150, 2016.

[13] K. Lu, Q. Zhou, R. Li, Z. Zhao, X. Chen, J. Wu, and H. Zhang, "Rethinking modern communication from semantic coding to semantic communication," *IEEE Wireless Communications*, 2022.

[14] F. Zhou, Y. Li, X. Zhang, Q. Wu, X. Lei, and R. Q. Hu, "Cognitive semantic communication systems driven by knowledge graph," *arXiv preprint arXiv:2202.11958*, 2022.

[15] P. Hagoort, G. Baggio, and R. M. Willems, "Semantic unification," in *The cognitive neurosciences, 4th ed.* MIT press, 2009, pp. 819–836.

[16] J. H. Siekmann, "Universal unification," in *International Conference on Automated Deduction*. Springer, 1984, pp. 1–42.

[17] D. B. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd, "Cyc: toward programs with common sense," *Communications of the ACM*, vol. 33, no. 8, pp. 30–49, 1990.

[18] S. L. Reed, D. B. Lenat *et al.*, "Mapping ontologies into cyc," in *AAAI 2002 Conference workshop on ontologies for the semantic Web*, 2002, pp. 1–6.

[19] Zbigniew Gomolkaa and Boguslaw Twaroga and Ewa Zeslawskaa and Ewa Dudek-Dyduchb, "Knowledge base component of intelligent almm system based on the ontology approach," *Expert Systems with Applications*, vol. 199, p. 116975, 2022.

[20] V. V. Gribova, A. S. Kleschev, F. M. Moskalenko, V. A. Timchenko, L. A. Fedorishchev, E. A. Shalfeeva, "Iacpaas cloud platform for developing shells of intelligent services: state and development prospects," *Programmnyye produkty i sistemy*, 2018.

[21] Filippov A. A., Moshkin V. S., Shalaev D. O., Yarushkina N. G., "Unified ontological data mining platform," *System Analysis and Applied Informatics*, pp. 77–82, 2016.

[22] Yu. A. Zagorulko, "Semantic technology for the development of intelligent systems, focused on domain experts," *Ontologiya proyektirovaniya*, pp. 30–44, 2015.

[23] Gulyakina N. A., Golenkov V. V., "Graphic-dynamic models of parallel knowledge processing: principles of construction, implementation and design," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, G. V.V., Ed. BSUIR, Minsk, 2012, pp. 23–52.

[24] C. W. Holsapple and K. D. Joshi, "A collaborative approach to ontology design," *Communications of the ACM*, vol. 45, no. 2, pp. 42–47, 2002.

[25] Ford, Brian and Schiano-Phan, Rosa and Vallejo, Juan, *Component Design*, 11 2019, pp. 160–174.

[26] D. Shunkevich, D. Koronchik, "Ontological approach to the development of a software model of a semantic computer based on the traditional computer architecture," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*. BSUIR, Minsk, 2021, pp. 75–92.

[27] C. Ballinger, "The teradata scalability story," *Technical report, Teradata Corporation*, 2009.

[28] "Cyc platform," 2022. [Online]. Available: https://cyc.com/platform/

[29] R. Gurunath and D. Samanta, "A novel approach for semantic web application in online education based on steganography," *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT)*, vol. 17, no. 4, pp. 1–13, 2022.

[30] Rudikova L. V., Zhavnerko E. V., "About data modeling of subject-domains of a practice-oriented orientation for a universal system for storing and processing data," *System Analysis and Applied Informatics*, pp. 4–11, 2017.

[31] J. Bai, L. Cao, S. Mosbach, J. Akroyd, A. A. Lapkin, and M. Kraft, "From platform to knowledge graph: evolution of laboratory automation," *JACS Au*, vol. 2, no. 2, pp. 292–309, 2022.

[32] Ian Robinson, Jim Webber and Emil Eifrem, *Graph databases*. O'Reilly Media, Inc., 2015.

[33] Abramsky Mikhail Mikhailovich, Timerkhanov Timur Ildarovich, "Comparative analysis of the use of relational and graph databases in the development of digital educational systems," in *Vestnik NGU*. Russian Federation, Novosibirsk, Vestnik NSU, 2018, pp. 5–11.

[34] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins, "A comparison of a graph database and a relational database: a data provenance perspective," in *Proceedings of the 48th annual Southeast regional conference*, 2010, pp. 1–6.

[35] Klimanskaya E.V., "Modern platforms for intelligent information processing: Graph databases," *Nauka vchera, segodnya, zavtra*, pp. 9–16, 2014.

[36] C. Chen *et al.*, "Multi-perspective evaluation of relational and graph databases," 2022.

[37] Amir Hosein Khasahmadi and Kaveh Hassani and Parsa Moradi and Leo Lee and Quaid Morris, "Memory-based graph networks," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=r1laNeBYPB

[38] O. P. Kuznecov, *Diskretnaya matematika dlya inzhenera: Uchebnik dlya vuzov [Discrete Mathematics for an Engineer: A Textbook for High Schools]*. Moscow: Lan', 2009.

[39] Reinhard Diestel, *Graph Theory*. Hamburg, Germany: Universität Hamburg, 2017.

[40] C. A. Sen, S. Parkkonen, Yu. A. Zobni, "Application of graph databases to form knowledge bases of complex systems," in *Problemy formirovaniya yedinogo prostranstva ekonomicheskogo i sotsial'nogo razvitiya stran SNG (SNG-2017)*. Tyumen: Tyumen Industrial University, 2017, pp. 165–172.

[41] A.N.Naumov, A.M.Vendrov, V.K.Ivanov, *Database and knowledge management systems*. M.: Finance and statistics, 1991.

[42] T. A. Gavrilova, V. F. Khoroshevsky, *Knowledge bases of intelligent systems*. SPb: Peter, 2000.

[43] A. A. Bashlykov, "Knowledge management systems," in *Avtomatizatsiya, telemekhanizatsiya i svyaz' v neftyanoy promyshlennosti*, 2010, pp. 33–39.

[44] ——, "Methodology for building knowledge base management systems for intelligent systems," in *Programmnyye produkty i sistemy*, 2013, pp. 131–137.

[45] V. Golenkov and N. Guliakina and I. Davydenko and A. Eremeev, "Methods and tools for ensuring compatibility of computer systems," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellek-*

tual'nykh system [Open semantic technologies for intelligent systems], V. Golenkov, Ed. BSUIR, Minsk, 2019, pp. 25–52.

[46] Golenkov V.V., Gulyakina N.A., Davydenko I.T., Shunkevich D.V., Eremeev A.P., "Ontological design of hybrid semantically compatible intelligent systems based on the semantic representation of knowledge," in *Ontologiya proyektirovaniya*, G. V.V., Ed. Russian Federation, Samara: Samara National Research University named after Academician S.P. Korolev, 2019, pp. 132–148.

[47] M. J. Jacobs, "A software development project ontology," Master's thesis, University of Twente, 2022.

[48] V.V. Gribova and A.S. Kleschev and D.A. Krylov and F.M. Moscalenko, "The basic technology development of intelligent services on cloud platform iacpaas. part 1. the development of a knowledge base and a solver of problems," *Software engineering*, no. 12, pp. 3–11, 2015.

[49] Yurii I. Molorodov, "Development of information system based on ontological design patterns," in *5th International Conference Information Technologies in Earth Sciences and Applications for Geology, Mining and Economy,*. Institute of Computational Technologies, Siberian Branch of the Russian Academy of Sciences, 2019.

[50] C. M. Zapata Jaramillo, G. L. Giraldo, and G. A. Urrego Giraldo, "Ontologies in software engineering: approaching two great knowledge areas," *Revista Ingenierías Universidad de Medellín*, vol. 9, no. 16, pp. 91–99, 2010.

[51] D. N. Koronchik, "Unificirovannye semanticheskie modeli pol'zovatel'skih interfejsov intellektual'nyh sistem i tekhnologiya ih komponentnogo proektirovaniya [Unified semantic models of user interface for intelligent systems and technology for their develop]," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2013, pp. 403–406.

[52] V. P. Ivashenko and N. L. Verenik and A. I. Girel' and E. N. Sejtkulov and M. M. Tatur, "Predstavlenie semanticheskih setej i algoritmy ih organizacii i semanticheskoj obrabotki na vychislitel'nyh sistemah s massovym parallelizmom [Semantic networks representation and algorithms for their organization and semantic processing on massively parallel computers]," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2015, pp. 133–140.

[53] E. Iotti, "An agent-oriented programming language for jade multi-agent systems," 2018.

[54] D. Shunkevich, "Agentno-orientirovannye reshateli zadach intellektual'nyh sistem [Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems]," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2018, pp. 119–132.

[55] (2022, Nov) Implementation of the sc-memory. [Online]. Available: https://github.com/ostis-ai/sc-machine

[56] R. Bayer and K. Unterauer, "Prefix b-trees," *ACM Transactions on Database Systems (TODS)*, vol. 2, no. 1, pp. 11–26, 1977.

[57] K. Tsuruta, D. Köppl, S. Kanda, Y. Nakashima, S. Inenaga, H. Bannai, and M. Takeda, "c-trie++: A dynamic trie tailored for fast prefix searches," *Information and Computation*, vol. 285, p. 104794, 2022.

[58] D. Belazzougui, P. Boldi, R. Pagh, and S. Vigna, "Fast prefix search in little space, with applications," in *European Symposium on Algorithms*. Springer, 2010, pp. 427–438.

[59] Bhumij Gupta1, Dr. M.P. Vani, "An overview of web sockets: The future of real-time communication," in *International Research Journal of Engineering and Technology (IRJET)*, 2018.

[60] A. A. Naik and M. R. Khare, "Study of "websocket protocol for real-time data transfer"," *International Reasearch Journal of Engineering and Technology*, 2020.

[61] M. Tomasetti, "An analysis of the performance of websockets in various programming languages and libraries," *Available at SSRN 3778525*, 2021.

[62] Q. Liu and X. Sun, "Research of web real-time communication based on web socket," 2012.

[63] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, "Foundations of json schema," in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 263–273.

[64] T. Marrs, *JSON at work: practical data integration for the web*. " O'Reilly Media, Inc.", 2017.

[65] Myers B.A., Rosson M.B., "Survey on user interface programming," in *Proceedings SIGCHI'92: Human Factors in Computing Systems*. Monterrey, CA, 1992, pp. 195–202.

[66] (2022, Niv) Implementation of the interpreter of user interface sc-models. [Online]. Available: https://github.com/ostis-ai/sc-web

[67] M. Sadouski, "Semantic-based design of an adaptive user interface," in *International Conference on Open Semantic Technologies for Intelligent Systems*. Springer, 2022, pp. 165–191.

[68] J. Kong, W. Zhang, N. Yu, and X. Xia, "Design of human-centric adaptive multimodal interfaces," *Int. J. Hum.-Comput. Stud.*, vol. 69, pp. 854–869, 12 2011.

[69] (2022, Nov) OSTIS Metasystem. [Online]. Available: https://ims.ostis.net

Программная платформа для интеллектуальных компьютерных систем нового поколения

Зотов Н.В.

Данная работа посвящена проблемам обеспечения проектирования семантически совместимых компьютерных систем и их независимости от реализации платформ проектирования таких систем. Работа показывает высокий уровень значимости проектирования и реализации платформ нового поколения, а также определяет решение задачи в виде проектирования и разработки универсальных интерпретаторов логико-семантических моделей систем по принципам, лежащим в основе Технологии OSTIS. Вторая часть работы отражает текущее состояние реализуемой платформы, приводит достоинства и недостатки реализуемых в ней компонентов, предлагает пути совершенствования платформы.

Received 28.10.2022

Problems and prospects of automating various types and fields of human activity with the help of next-generation intelligent computer systems

Vladimir Golenkov
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
Email: golen@bsuir.by

Valery B. Taranchuk
Department of Computer Applications and Systems Belarusian State University
Minsk, Republic of Belarus
taranchuk@bsu.by

Mikhail Kovalev
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
Email: michail.kovalev7@gmail.com

***Abstract*—In the article, the principles for automating various fields of human activity using next-generation intelligent computer systems are considered. An ontology of various types of activities and related technologies is proposed. The specification of these principles is carried out on the example of human activity in the field of Artificial Intelligence.**

***Keywords*—OSTIS, ostis-system, activity, ontological approach, intelligent computer system, interoperability, knowledge base, SC-code**

I. Introduction

The key problem of the modern level of complex automation of *human activity* is as follows. Currently, either full automation of some classes of actions initiated by the corresponding teams is being carried out, or partial automation of some types of *human activity*, within which a person controls the appropriate automation tools. At the same time, automation of complex problems solving, which are reduced to several partially automated subproblemss, requires "manual" (non-automated) control of several automation tools simultaneously.

The principles underlying the transition to a higher level of automation of *human activity* are as follows. All automation tools (all services) currently managed by people are being "managed" by ***interoperable intelligent computer systems*** that are able to effectively interact with each other and, accordingly, are able to fully automate the solution of complex problems requiring the use of several automation tools (services) [1], [2], [3], [4], [5], [6].

In this paper, we will first clarify the basic concepts we use to consider the structure of *human activity*, then we will consider in detail the structure of the current state and the problems of *Human activity development in the field of Artificial Intelligence*. Next, we will generalize the principles of organization and automation of *Human activity in the field of Artificial Intelligence* to all the variety of types and directions of *human activity*.

II. Basic concepts underlying the formal description of the structure of human activity

activity
:= [process of situational impact on some dynamic system, aimed either at creating this system, or maintaining certain characteristics of this system, or its destruction, or its development (improvement)]
:= [system of all actions performed by some individual or collective entity or an integral subsystem of such actions corresponding to the purpose (duties) of this entity]
⊂ *process*

should be distinguished*
∋ {• *activity*
• *action*
}
⇒ *comparison**:
[Each *action* necessarily corresponds to the formulation of the problem that is solved as a result of performing this action. In this case, the *action* can be complex (non-atomic), i.e. it can be a hierarchical system of sub-actions, providing the solution of *sub-problemss* of the initial *problem*.

In contrast, each *activity* can correspond to several initial problems that are not sub-problems of other problems within this *activity*.

An example of an activity that is not an action is an activity carried out by some subject within some *field* (meaning the *activity* to solve all kinds of problems of the most different kinds that can be formulated within the specified *field*).]
⇒ *note**:
[Each activity and, accordingly, each action uniquely corresponds to a subject performing that activity.]

human activity
:= [a system of actions performed by people or human communities either "manually" or with "passive" tools (sticks, ropes, shovels, axes and e.t.c.) or with "active" tools (vehicles, excavators, chainsaws and e.t.c.)]

action
:= [process of changing the state of some dynamic system (from a given state to a desired target state), initiated and possibly directly carried out by some subject with the possible use of some tools, additional materials and information resources (in particular methods)]
:= [a purposeful (conscious) process of some subject acting on some object]
⇒ *note**:
[The subject of the action can be either individual or collective. The object of impact may have any complex structure and consist of any number of components that are impacted. The initiation and execution of an action can be carried out by different subjects using various auxiliary tools]
. ⊃ *information action*
:= [information process performed by some subject (including a computer processor)]
:= [process of changing the state of some information resource]

should be distinguished*
∋ {• *action*
• ***problem***
:= [specification of some action that contains enough information to perform that action]
:= [problem formulation]
⊃ *declarative problem formulation*
⊃ *procedural problem statement*
}

relationship defined on the action set
∋ *action**
:= [be an action performed to solve a given problem]
⇔ *inverse relation**:
*problem**
:= [to be a problem performed as a result of this action]

relationship defined on the set of activities
∋ *activity object**
:= [being an object over which a given activity is performed*]
∋ *product of activity**
:= [to be the product of a given activity*]
∋ *part of the time of existence**
:= [to be a segment of time of existence of a given temporal entity*]
∋ *activities whose object classes coincide**
∋ *activities performed simultaneously**

activity type
:= [class of activities]
:= [class of similar activities, which can be matched with a common technology that ensures the performance of all activities of this class]
:= [a class whose instances (elements) are equivalent (similar) activities performed in general by different cybernetic systems]
∋ *example'*:
design
:= [design activity]
:= [human activity whose product is the design documentation of some created entity]
:= [building a complete specification (design documentation) of some entity to be created]
⇒ *note**:
[The products of this activity are the specifications of any socially significant created entities]
⇒ *particular activity over a subclass of activity objects**:
- *ostis-systems design*
- *circuit design*
- *building design*

∋ *example'*:
life cycle support
⇒ *particular activity over a subclass of activity objects**:
- *chip life cycle support*
- *buildings life cycle support*
- *support of life cycle of ostis-systems*
  ⇒ *particular activity over a subclass of activity objects**:
  - *support of the ostis-systems knowledge base life cycle*
  - *support for the life cycle of ostis-systems problem solvers*
  - *support life cycle of ostis-systems interfaces*

∋ *human resources training*
:= [educational activity]
∋ *production*
∋ *environmental activity*
∋ *construction activity*
∋ *health care activity*
∋ *administrative activity*
∋ *research activity*

should be distinguished*
∋ {• *type of activity*
• *class of actions*
}

should be distinguished*
∋ {• *class of actions*
• *class of problems*
:= [a set of similar (similar) formulations of specific problems that can easily be generalized by replacing some constants going into those formulations with variables]
• *problem class formulation*
:= [a generalized problem class formulation (specification) that "turns" into a specific problem formulation when specific values are assigned to all the free variables included in this generalized problem class formulation]
}

relationship defined on the action set
∋ *method**
:= [to be a method that provides all actions of a given class of actions, or solving all problems of a given class of problems, or performing a particular given action, or solving a particular given problem*]
∋ *maximal action class**
:= [to be the maximum class of actions performed by a given method*]

method
:= [information construct whose interpretation by any subject belonging to the corresponding class of subjects ensures that any action belonging to the corresponding class of actions]
⊃ *methodology*
:= [method implemented by a person or group of people]

a relation defined on a set of activities

∋ *technology**

:= [be a technology that ensures the performance of each activity belonging to a given type of activity*]

∋ *class of objects of activity**

:= [be a class of activity objects for a given type of activity*]

∋ *activity product class**

:= [be a class of activity products for a given type of activity*]

∋ *a particular type of activity over a subclass of objects of activity**

:= [a particular type of activity, the class of objects of which is a subclass of objects of activity of a given type of activity*]

∋ *a particular type of activity over a class of components of objects of activity**

:= [a particular type of activity, the class of objects of which is the class of components of the objects of activity of a given type of activity*]

∋ *a particular type of simultaneously performed activity**

:= [a particular type of activity, each of which is performed simultaneously (in parallel) with activities belonging to a given type of activity*]

∋ *a particular type of activity performed at some stage**

:= [a particular type of activity, each of which is performed as one of the stages of the activity, belonging to a given type of activity*]

∋ *types of activities whose object classes coincide**

∋ *types of activities performed by the same entities**

technology

:= [a system of knowledge, skills and tools that ensure the performance of each activity of the appropriate type by the relevant subjects]

⇒ *note**:

[The basis of any *technology* is multiplicity - either the multiplicity of creating similar (analogous) entities, or the multiplicity of performing similar actions, the multiplicity of using similar methods (techniques). Obviously, the higher the degree of similarity (proximity, convergence) of repeatedly created entities and repeatedly used methods, the higher their unification, the simpler the corresponding *technology* will be.]

scope of execution*

:= [be a subject domain and a corresponding ontology (possibly with some child subject domains and corresponding ontologies) containing knowledge sufficient to perform a given action, or a given class of actions, or a given activity, or a given type of activity*]

⊃ *scope of the action**

⊃ *scope of the action class execution**

⊃ *scope of activity**

⊃ *scope of the type of activity**

subject*

:= [be a subject performing a given action or a given activity*]

⇒ *note**:

[The initiation of an action or activity performed by a subject can be carried out either independently (on its own initiative), or on the initiative (command, task) of another subject.]

information resource

:= [a socially significant information structure that is a product of relevant human activity, which requires not only its creation, but also maintenance (updating, updating)]

⊃ *project documentation*

⊃ *scientific theory*

:= [the product of scientific research activity, which is a strict description of the properties and patterns of a certain class of entities]

⊃ *standard*

⊃ *specification of a set of techniques of the corresponding technology*

⊃ *specification of tools of the corresponding technology*

project documentation

:= [full specification of the entity being created]

:= [product of project activity]

:= ["digital" copy of some entity]

:= [an information model (description) of an entity that has sufficient completeness (detail) to reproduce (reproduce) this entity]

III. Problems and prospects of human activity development in the field of Artificial intelligence

Earlier in the work [7] we clarified:

- architecture of *next-generation intelligent computer systems*
- how the activity (functioning) of *next-generation intelligent computer system* is carried out
- the way *applied engineering human activity for designing new-generation intelligent computer systems* is automated and supporting all subsequent stages of their life cycle.

Let's clarify how the whole complex of *human activity in the field of Artificial Intelligence* is carried out and automated.

A. General assessment of the current state of human activity in the field of Artificial intelligence

Let's consider the need to move the organization of *human activity* of ***Artificial intelligence*** to a fundamentally new level, ensuring the formation of a market for **semantically compatible *new-generation intelligent computer systems***, developed on the basis of a fundamentally new complex of *semantically compatible* ***Artificial intelligence technologies***.

Now it is important to investigate not only *models for solving intelligent problems* in intelligent computer systems of various types, but also methodological problems of the current state of *Artificial intelligence* in general and ways to solve these problems.

An analysis of the current state of work in the field of *Artificial intelligence* shows that this scientific and technical discipline is in a serious methodological crisis. Therefore, it is necessary:

- Identification of the main causes of this crisis;
- Clarification of the main measures aimed at its elimination.

The solution of the crisis problems under consideration requires:

- An essential fundamental system-wide rethinking of *everything* we do in the field of *Artificial intelligence* and *how* we do it, i.e. requires clarification of the characteristics of *intelligent computer systems*, clarification of the concept of a community consisting of *intelligent computer systems* and users interacting with them, clarification of the requirements for *intelligent computer systems*, as well as clarification of methods and means of their creation and use.
- The realization that *Cybernetics, Computer Science* and *Artificial intelligence* are a common fundamental science that requires an integrated approach to the construction of general formal models of systems based on information processing (*cybernetic systems*) by *convergence* and *integration* of formal models of various components of these systems [8]. Thus, the current stage of the development of *Artificial intelligence* is a transition from the accumulated to the current moment variety of models for solving various types of problems to the transformation of this variety into a coherent system of *semantically compatible* models;
- The realization that now it is necessary not to expand the diversity of points of view, but to learn to coordinate them, to ensure their *semantic compatibility*, improving the appropriate methods.

Discussing the modern problems of *convergence* of various models in the field of *Artificial intelligence* and the construction of integrated *hybrid models*, it is appropriate to recall «the fantastic story of D.A. Pospelov ”Contact”, dedicated to the contact of different worlds. In it, the main character popularly expounds his theory of *conceptual faults* <...>. This theory resembles the history of a long period of differentiation of sciences, when various scientific disciplines developed independently, like parallel worlds, only occasionally touching each other, and individual scientists, receiving an increasingly narrow specialization, knew little about the achievements of even their "close brethren". Fortunately, in recent years, new directions of contact between individual disciplines have been emerging more and more often, ideas are interpenetrating, analogies are being established between the results obtained and development trends. This is largely due to the emergence and widespread introduction of advanced information and communication technologies into all spheres of society <...>. Modern technologies rely on the achievements of many scientific and technical disciplines, among which new-generation synthetic sciences - the sciences of artificial» come to the fore.

⇐ *quote**:
Tarasov V.B. fMStIO – 2002bk/p.13 [9]

Analyzing the current state of work in the field of *Artificial intelligence (AI)*, it should be stated that the *conceptual gap* between the various directions of *Artificial intelligence* is an obvious fact. This is confirmed by the following quote from the book by V.B. Tarasov [9] «again, as at the dawn of AI, the formation of unified methodological foundations of AI, the development of theoretical problems of creating intelligent systems of new generations, the development of unconventional hardware and software are becoming relevant. Here, great prospects are associated with the use of ideas and principles of synergetics in AI. The term "synergetics" itself comes from the word "synergy", meaning joint action, cooperation. According to the "father of synergetics" G. Haken, such a name is quite suitable for the modern theory of complex self-organizing systems for two reasons: a) the joint actions of many elements of a developing system are being investigated; b) to find common principles of self-organization, it requires the combined efforts of representatives of various disciplines».

⇐ *quote**:
Tarasov V.B. fMStIO – 2002bk/p.14 [9]

In order to make sure that there is a *conceptual rift* between different directions of *Artificial intelligence*, it is enough to simply list the main directions of work of conferences on the subject of *Artificial intelligence*, paying attention to the fact that many of them are developing independently of others:

- synergetic models of self-organization of intelligent computer systems;
- hybrid intelligent computer systems;
- collaborative intelligent computer systems;
- soft computing, intelligent computing;
- modeling of non-factors;
- non-classical, multivalued, modal, pseudophysical, inductive, fuzzy logic and approximate reasoning, logic programs;
- fuzzy sets, relations, graphs, algorithms;
- functional programs, fuzzy algorithms, genetic algorithms, production models;
- neural network models;
- parallel asynchronous models of decentralized problem solving;
- signal processing;
- multisensory convergence, sensorimotor coordination;
- situational management models.

Overcoming the *conceptual rift* between different directions of research in the field of *Artificial intelligence* is a kind of "leap" across the "conceptual abyss", which requires special concentration of efforts. You can’t jump over the abyss in two jumps.

If we briefly characterize the **current state** of all work in the field of ***Artificial Intelligence***, it is an **illusion of well-being**. There is an active local development of various directions of *Artificial Intelligence* (*non-classical logics*, *formal ontologies*, *artificial neural* *networks*, *machine learning*, *soft computing*, *multi-agent systems*, etc.), but there is no comprehensive increase in the level of *intelligence* of modern *intelligent computer systems*. This first of all requires the convergence and *integration* of all directions of *Artificial intelligence* and the corresponding building of a ***general formal theory of intelligent computer systems***, and also the transformation of the modern variety of *development tools* (frameworks) of various components of *intelligent computer systems* into a single ***Technology of integrated design and support the whole life cycle of intelligent computer systems***, which guarantees the compatibility of all developed components of *intelligent computer systems*, as well as the compatibility of *intelligent computer systems* themselves as independent entities (agents, actors) interacting with each other within complex systems of automation of complex collective *human activity* (smart houses, smart hospitals, smart schools, smart manufacturing enterprises, smart cities, etc.). Thus the epigraph of the current state of work in the field of Artificial Intelligence is the well-known statement from Ecclesiastes: ”A time to scatter stones and a time to gather stones - to all in good time”.

«Unfortunately, in today’s discussions on AI (Artificial Intelligence) scientific debates are often substituted by exaggerated expectations from the rapid introduction of AI and a significant narrowing of the topic of AI, which has been reduced only to *machine learning* based on *artificial neural networks*. <...> Meanwhile *ontologies*, *knowledge bases*, *methods of reasoning* and *decision making*, *methods of synthesis and analysis of complex structures*, intelligent cyber-physical systems, *digital twins*, *autonomous systems*, systems of analysis of both "big" and "small" data were left out of the National Strategy. <...>

Recognizing the importance of *machine learning* based on *artificial neural networks*, world-class scientific and practical results should be sought at the intersection of different disciplines in the ***convergence*** of different AI technologies and ***integration*** of multidisciplinary *knowledge*. In this regard, *knowledge* formalization in the form of *ontologies* and *knowledge bases* within *Semantic Web* is seen as one of the fundamental directions for the creation of AI. Indeed, what kind of *intelligence* can there be without using the *knowledge* of modern textbooks, on the basis of which AI will understand the *context of the situation*, draw *conclusions* and *make decisions*? <...>

Another key direction of AI that is not reflected in the Russian AI strategy is *distributed decision-making*, which is increasingly becoming collective for the rapidly developing smart Internet of Things and autonomous control systems, starting with unmanned cars, aircraft, ships, etc.

Gartner has declared 2020 the year of "autonomous things" which, according to the company, have already evolved from "digital" to "smart". In the next phase, autonomous things with their own AI are expected to "talk" to each other and the scientific agenda will include ***semantic interoperability*** of AI systems that will not only exchange data, but also negotiate to agree on solutions. The U.S. AI research roadmap highlights such direction as the *connectivity* of *Artificial intelligence* systems (Integrated Intelligence) and their *meaningful interaction*, along with various types of *Self-Aware Learning* in systems, as key.»

⇐ *quote**:

Barinov I.I.. DevelSFoftheConAI- 2021art/pp. 264-265 [10]

The **key reason** for the **methodological problems** of the current state of *Artificial intelligence* and a serious challenge for specialists in this field is the curse of the ***Babylonian Pillar*** of [11], which haunts us at all levels:

- at the level of internal organization of *problem solving* in *intelligent computer systems*;
- at the level of interaction of *intelligent computer systems*, both among themselves and with users;
- at the level of interaction of scientists working in the field of *Artificial intelligence*, which prevents the creation of a *general formal theory and standard of intelligent computer systems*, as well as the *Technology of integrated design and support of the entire life cycle of intelligent computer systems*
- at the level of interaction between scientists, engineers who develop applied *intelligent computer systems*, university professors who train specialists in the field of *Artificial intelligence*, as well as students, undergraduates and graduate students.

The complexity of currently developed *intelligent computer systems* and Artificial Intelligence technologies has reached such a level that their development requires not just large creative teams, but also a significant increase in the qualifications and quality of these teams. It is well known that the qualification of a team of developers is determined not only by the qualification of its members, but also by the efficiency and atmosphere of their interaction. It is also known that the quality of any technical system is a reflection of the quality of the team that developed the system. Can a team of sufficiently qualified specialists, many of whom are not highly interoperable, develop an intelligent computer system with a high level of interoperability, much less a technology for integrated support of the entire life cycle of intelligent computer systems of this level? The obvious answer to this question and the obvious complexity of creating workable creative teams indicate the main challenge addressed to Artificial Intelligence specialists at the present time. Thus, the requirements for the new-generation intelligent computer systems, determining their ability to individually and collectively solve complex problems, should also be imposed on the developers of these systems, as well as the developers of any other complex objects, since all complex types and directions of human activity are collective and creative.

Creating a rapidly growing market of semantically compatible *intelligent computer systems* is the main goal addressed to the experts in the field of Artificial Intelligence, requiring overcoming the ***Babylonian pandemonium*** in all its manifestations, forming a high culture of agreement and a unified, coordinated form of representation of collectively accumulated, improved and used knowledge. Scientists, working in the field of *Artificial Intelligence*, should ensure the ***convergence*** of the results of different directions of *Artificial intelligence* and build a *general formal theory of intelligent computer systems*, as well as the *Integrated Design Technology for semantically compatible intelligent computer systems*, including appropriate standards for *intelligent computer systems* and their components. Engineers *who develop applied intelligent computer systems* should collaborate with scientists and participate in the development of the *Comprehensive technology for fesign of semantically interoperable intelligent computer systems*, and support all subsequent stages of the life cycle of these systems.

The isolation of various research directions in the field of *Artificial Inteiligence* is the main obstacle to the creation of a *comprehensive technology for the design of semantically compatible intelligent computer systems*, as well as the *Technology of integrated support* for all subsequent stages of the life cycle of *intelligent computer systems*.

B. Structure of activity in the field of Artificial Intelligence

In order to consider the problems of further development of *activity* in the field of *Artificial Intelligence* and, in particular, the problems of complex automation of this *activity*, it is necessary to specify the structure of the mentioned *activity*.

Human activity in the field of *Artificial Intelligence* are aimed at research and creation of *intelligent computer systems* of various kinds and different purposes. The objects of research in *Artificial Intelligence* are:

- *individual intelligent computer systems* (in particular, cognitive agents);
- *multiagent intelligent computer systems* (in particular, communities consisting of *individual intelligent computer systems*);
- Human-machine communities consisting of *intelligent computer systems* and their users.

The main goals of human activity in *Artificial Intelligence* are:

- building a formal theory of *intelligent computer systems* (artificial *intelligent systems*);
- creation of technologies (techniques and tools) that provide *design, implementation, maintenance and operation of intelligent computer systems*;
- transition to a fundamentally new level of complex automation of all types of human activity, which is based on mass application of *intelligent computer systems* and which implies:
  - not only the presence of *intelligent computer systems* capable of understanding each other and coordinating their activities,
  - but also consideration of the general structure of *human activity* carried out under the conditions of its new level of automation (smart-society activity), which should be "understandable" to the used *intelligent computer systems* and which will require a substantial rethinking of the modern organization of *human activity*.

Artificial Intelligence as a *field* of *human activity* includes the following *activities*:

- ***Research activity in the field of Artificial Intelligence***, in the process of which there is a competition of different points of view and approaches to the construction of formal models of various components of *intelligent computer systems*. The ultimate goal of such activity is the constantly evolving *General theory of intelligent computer systems*. The objects of research of this theory are *intelligent computer systems* and their formal *logical-semantic models*. These models include formal models of various types of *knowledge*, which are part of *knowledge bases* of intelligent computer systems, as well as various *problem-solving models* (logical models of various types, neural network models, genetic models, productive models, functional models, etc.).
- ***The development of the Intelligent Computer Systems Standard***, which includes the permanent evolution of this standard and maintains the integrity of each version of it. The current version of the *Standard for Intelligent Computer Systems* is the consensus (generally accepted) currently part of the *General Theory of Intelligent Computer Systems*.
- ***Development of technology for designing intelligent computer systems***, which includes a family of design methodologies, as well as methods and tools for automating *design* of various *components* of *intelligent computer systems* and *intelligent computer systems* in general. The result of designing *intelligent computer systems* is a complete formal logical-semantic model of this system.
- ***The development of technology for the implementation of designed intelligent computer systems***, as well as technologies for the operation and maintenance of *intelligent computer systems*. In the basis of the technology of implementation (production) of the designed *intelligent computer systems* lies the *universal interpreter of formal logical-semantic models of intelligent computer systems*, which are the result of the design of the specified systems. The

specified universal interpreter can be implemented either as a *software system* on modern computers, or as a *universal new-generation computer*, oriented to the interpretation of formal *logical-semantic models of intelligent computer systems*.

- ***Applied engineering activity in the field of Artificial Intelligence***, i.e., the direct design, implementation and maintenance, which includes updating (re-engineering) performed during operation, of specific *intelligent computer systems.*
- ***Training activity in the field of Artificial Intelligence*** are aimed at training specialists in the field of *Artificial Intelligence* and at continuous professional development of existing specialists in this field. Without effective organization of training activities in the field of *Artificial Intelligence*, rapid progress in this field is impossible. Direct inclusion of learning activity in the general structure of human activity in the field of *Artificial Intelligence* is caused by the following circumstances:
  - the necessity of deep convergence between different fields and types of activities in the field of Artificial Intelligence and the corresponding specificity of requirements to specialists in this field – each such specialist must be competent enough both in research activities in the field of Artificial Intelligence, and in the development of technologies (methods and means) of designing intelligent computer systems, and in the development of technologies of reproduction (realization) of the designed *Artificial Intelligence*;
  - high rate of evolution of results in the field of *Artificial Intelligence*, which makes it necessary to organize the training of relevant specialists by connecting them directly not to training (simplified) projects, but to real projects, implemented at the moment. Otherwise, trained specialists will have the qualification of "yesterday's day";
  - the significant expansion of the volume of work in the field of *Artificial Intelligence* and the urgent need for mass training of relevant specialists.

  The difficulty of ***Training of young professionals in the field of Artificial Intelligence*** lies not only in the high degree of scientific intensity of this field, but also in the fact that the formation of relevant knowledge and skills in them is carried out in conditions of rapid moral aging of the current state of *Artificial Intelligence* technology, significant changes in which occur during the education of students and undergraduates. Therefore, it is necessary to teach not the current level of development of *Artificial Intelligence*, but the level of development that will be achieved in five years or more.

  When training young professionals in the field of *Artificial Intelligence*, it is necessary to form in them:
  - formalization culture (mathematical culture);
  - systemic culture (in particular, the ability to perform qualitative stratification of complex dynamic systems);
  - technological culture (in particular, the ability to distinguish between what should be unified and what unification limits the direction of evolution of a given class of complex systems);
  - technological discipline;
  - culture of collective creativity (in particular, initial *interoperability*);
  - high *cognitive activity* and motivation;
  - ability to combine individual creative freedom and independence with ensuring compatibility of one's results with the results of one's colleagues, i.e. to combine freedom in creating (generating) new meanings with the coherence (compatibility) of forms of their representation – notions, terms and syntax are not argued about, but agreed upon.
- ***Organizational activity in the field of Artificial Intelligence***, aimed at creating an infrastructure for the quality performance of all other activities in the field of *Artificial Intelligence*, namely:
  - to ensure a deep *convergence* between the different fields and activities in the field of *Artificial Intelligence* and, in particular, between theory, technology and engineering practice in this field;
  - to balance tactics and strategy in the development of *Artificial Intelligence* activities as a key basis for significantly increasing the level of automation of all types of *human activities* and the transition to *smart society*.

The considered decomposition of *human activity* in the field of *Artificial Intelligence* by *types of activities* is not a traditional feature of *scientific-technical disciplines* decomposition. Usually, the decomposition of *scientific-technical disciplines* is carried out according to the content directions which correspond to the decomposition of *technical systems* studied and developed within these *scientific-technical disciplines*, i.e. correspond to the allocation of various kinds of components in these *technical systems*. For *Artificial Intelligence* such directions are:

- Research and development of formal models and knowledge representation languages;
- research and development of knowledge bases;
- research and development of logical models of knowledge processing;
- research and development of artificial neural networks;
- research and development of computer vision subsystems;
- research and development of subsystems for processing natural language texts (syntactic analysis, comprehension, synthesis);
- and many others.

The importance of decomposition of *Artificial Intelligence* by *types of activities* is determined by the fact that the allocation of different *types of activities* allows to clearly set a problem for the development of automation tools for these *types of activities*.

Here is the general structure of *human activities in the field of Artificial Intelligence*.

Human activities in Artificial Intelligence
:= [Artificial Intelligence (as a scientific and technical discipline]
∈ *scientific-technical discipline*
:= [human activity in the *Subject domain of intelligent computer systems*]
∈ *activity*
⇒ *decomposition**:
{
- *Integral activity of support the life cycle of all types of intelligent computer systems*
  ⇒ *decomposition**:
  support the life cycle of intelligent computer systems
- *Support the life cycle of the General Theory of Intelligent Computer Systems*
  ∈ *research activity*
- *Smart computer systems standard life cycle support*
  ∈ *standardization*
  ⇒ *part**:
  Support the life cycle of the ostis-systems Standard
- *life cycle Support for Integrated Life Cycle Support Technologies for Intelligent Computer Systems*
  ∈ *support technology life cycle*
  := [technology creation and maintenance]
  ⇒ *part**:
  OSTIS Technology life cycle Support
- *Support of the human resources life cycle for Human Activity in the field of Artificial Intelligence*
- *Support the life cycle of the system of complex organization of interaction between all directions of Human Activity in the field of Artificial Intelligence*
  ∈ *supporting the life cycle of meta-systems of complex management support and providing life cycle support for entities of the respective class*

}

support the life cycle of intelligent computer systems
∈ *type of activity*
⇒ *generalized decomposition**:
{
- *designing intelligent computer systems*
- *production of intelligent computer systems*
- *initial training of intelligent computer systems*
- *quality monitoring of intelligent computer systems*
- *recovery of the required quality level of intelligent computer systems*
- *reengineering intelligent computer systems*
- *security assurance of intelligent computer systems*
- *operation of intelligent computer systems by end users*

}

Technology of intelligent computer systems life cycle support technology
⇒ *type of activity**:
smart computer systems life cycle support
⇒ *decomposition**:
{
- *Technology of intelligent computer system design*
  ⇒ *type of activity**:
  designing intelligent computer systems
- *Technology of production intelligent computer systems*
  ⇒ *type of activity**:
  intelligent computer system production
- *Technology for initial training of intelligent computer systems (activity-specific adaptation)*
  ⇒ *type of activity**:
  initial training of intelligent computer systems
- *Technology of Quality monitoring for intelligent computer systems*
  ⇒ *type of activity**:
  quality monitoring of intelligent computer systems
  := [planned testing and diagnosis of intelligent computer systems]
- *Technology of restoring the required level of quality of intelligent computer systems during their operation*
  := [Technology for detecting and correcting potentially dangerous situations and events in the operation of intelligent computer systems (errors, inconsistencies, etc.)]
  ⇒ *type of activity**:
  restoring the required level of quality of intelligent computer systems
- *Technology of reengineering intelligent computer systems*
  := [Technology of improving, modernizing, updating intelligent computer systems]
  ⇒ *type of activity**:
  reengineering intelligent computer systems
- *Technology of intelligent computer systems security*
  ⇒ *type of activity**:
  securing intelligent computer systems
- *Technology of operation of intelligent computer systems by end users*
  ⇒ *type of activity**:
  exploitation of intelligent computer systems by end users

}

C. Current state and current problems of Artificial Intelligence

Let's consider in which directions the evolution (quality improvement) of *Artificial Intelligence* activities should take place, as well as the evolution of the products of these activities.

1) Current state and current problems of ***research activities in the field Artificial Intelligence****:* Currently, research in *Artificial Intelligence* is actively developing in a wide range of different directions (*knowledge representation models*, different kinds of *logics* – deductive, inductive, abductive, clear, fuzzy, various kinds of *artificial neural networks*, machine learning, decision making, goal setting, behavior planning, situational behavior, multi-agent systems, computer vision, recognition, data mining, soft computing, and more). However:

- There is no consistency of *definitions* systems in different directions of *Artificial Intelligence* and, as a consequence, there is no *semantic compatibility* and *convergence* of these directions, resulting in significant difficulties in the direction of building

General theory of intelligent systems with a high level of formalization. The existence and continuing increase in the "height of barriers" between different research directions in the field of *Artificial Intelligence* is manifested in the fact that a specialist working within a particular direction of *Artificial Intelligence*, attending meetings of "not his" section at a conference on *Artificial Intelligence*, can understand little there and, accordingly, learn something useful for himself;
- There is a lack of motivation and awareness of the urgent need for mentioned *convergence* between different directions of *Artificial Intelligence*;
- There is no real movement in the direction of building *General theory of intelligent systems*, because there is no appropriate motivation and awareness of the acute practical need for it;
- There is no rigorous and consistent specification of the concept of *intelligent computer system*. So far, the Turing Test has been used for this purpose. A superficial interpretation of the Turing Test has given rise to various imitations of intelligence in the style of "small talk". In fact, a meaningful, goal-oriented dialogue should be taken into account, in which the intelligence of *intelligent computer system* is defined as its non-trivial contribution to the collective solution of some intelligent (creative) problem.

2) Current status of ***intelligent computer systems Standard****:* Currently, the need for unification and standardization of *intelligent computer systems* is not realized, which significantly hinders the creation of *complex technology of Artificial Intelligence*.

3) Current state and current problems of ***development****:* ***technologies of intelligent computer system design***

Modern *technology of Artificial Intelligence* is a whole family of all sorts of private technologies focused on development various kinds of *intelligent computer systems* components that implement a wide variety of information representation and processing models, as well as focused on development different classes of *intelligent computer systems*. However:
- high complexity of development of *intelligent computer systems*;
- high qualification of developers is required;
- modern *technologies of Artificial Intelligence* do not fundamentally ensure the development of such *intelligent computer systems*, which eliminate the drawbacks of modern *intelligent computer systems* and, in particular, provide a sufficiently high level of interoperability;
- compatibility of *design technologies of different classes of AI intelligent computer systems* is practically absent and, as a consequence, there is no *semantic compatibility* and interaction of the developed *intelligent computer systems*, so the system integration of *intelligent computer systems* is done manually;
- there is no *complex technology of intelligent computer systems design*;
- there is no compatibility between existing *particular design technologies for various components of intelligent computer systems* (knowledge bases, problem solvers, intelligent interfaces). There are tools for component development, but it is necessary to "glue" (connect, integrate) developed components manually, because there are no comprehensive tools to develop *intelligent computer systems* as a whole.

4) The current state and current problems of ***development technologies for the implementation of designed intelligent computer systems*** *as well as their operation and maintenance:* There have been a number of attempts to develop *new-generation computers* focused on the use of *in intelligent computer systems*. But all of them were unsuccessful because they were not oriented towards the whole variety of *problem-solving models* in *intelligent computer systems*. In this sense, they were not *universal computers* for *intelligent computer systems*.

Developed *intelligent computer systems* can use a variety of combinations of *intelligent problem-solving models* (logic models corresponding to various kinds of logics, neural network models of various kinds, goal-setting models, plan synthesis, complex object control models, natural language text understanding and synthesis models, etc.). However, modern traditional (von Neumann's) *computers* are not able to interpret all the variety of these *problem-solving models* in a sufficiently productive way. At the same time, the development of specialized *computers* focused on the interpretation of any one *problem-solving model* (neural network model or any logical model) does not solve the problem, because in *intelligent computer system* several different *problem-solving models* must be used at once, and in various combinations.

Currently, there is no comprehensive approach to the technological support of all stages of the *intelligent computer systems* life cycle – not only to support the design and implementation (assembly, production) of *intelligent computer systems*, but also to the technological support of maintenance, re-engineering and operation of *intelligent computer systems*.

The semantic unfriendliness of the *user interface* and the lack of built-in intelligent help systems that allow you to query information about interface elements and system features leads to low operational efficiency of all *intelligent computer system* features.

5) Current state and current problems of ***applied engineering in the field of Artificial Intelligence****:* We have accumulated quite a lot of experience in the development of *intelligent computer systems* for various purposes - systems of medical diagnostics automation, as well as diagnostics of complex technical systems, intelligent learning, information and help systems, systems of natural language communication, intelligent computer personal assistants, intelligent corporate systems, intelligent systems of situational management of various kinds of complex objects, systems of intelligent analysis of big data. However:

- The level of efficiency of practical use of scientific results in the field of *Artificial Intelligence* clearly does not correspond to the current level of development of these scientific results themselves. In order to improve the level of effectiveness of the practical use of the mentioned scientific results, collaborative efforts of scientists creating new models of intelligent problem solving, developers of design and implementation technologies, and developers of applied *intelligent computer systems* are required.
- There is no clear systematization of the variety of *intelligent computer systems*, corresponding to the systematization of automated *types of human activity*;
- There is no *convergence of* ***intelligent computer systems*** that provide automation of *fields of human activity* belonging to the same *type of human activity*;
- There is a lack of *semantic compatibility* (semantic unification, mutual understanding) between *intelligent computer systems*, the main reason being the absence of a concerted system of common *concepts* used;
- Analysis of the problems of complex automation of all *types of human activity* convinces us that further *automation of human activity* requires not only increasing the level of *intelligence* of the corresponding *intelligent computer* systems, but also to substantially increase their ability level:
  - build its *semantic compatibility* (understanding) both with other *computer systems* and with its users;
  - maintain this *semantic compatibility* in its own evolution, as well as the evolution of users and other *computer systems*;
  - coordinate with users and other *computer systems* in the collective solution of various problems;
  - participate in the distribution of work (subproblems) in the collective solution of various problemss.

It is important to emphasize that the implementation of the above capabilities will create the possibility for substantial and even complete automation of *system integration of computer systems* into complexes of interacting *intelligent computer systems* and automation of reengineering of such complexes. Such automation of system integration and its reengineering:

- will give the complexes of computer systems the opportunity to adapt independently to the solution of new problems;
- will significantly increase the efficiency of operation of such complexes of computer systems, as the reengineering of system integration of computer systems included in such a complex is often in demand (for example, in the reconstruction of enterprises);
- significantly reduces the number of errors compared to "manual" (non-automated) execution of *system integration* and its *reengineering*, which, in addition, require high skills.

Thus, the next stage of increasing automation of *human activity* urgently requires the creation of such *intelligent computer systems*, which could by themselves (without a system integrator) combine to solve complex problems together.

6) Current state and current problems of ***academic activities in the field of Artificial Intelligence****:* Many leading universities are training specialists in *Artificial Intelligence*. At the same time it is necessary to note the following features and problems of the current state of this activity:

- Since *activities in the field of Artificial Intelligence* combines both a high degree of science-intensive and a high degree of engineering complexity, training specialists in this field requires simultaneous formation of both research skills and knowledge and engineering-practical skills and knowledge, as well as system and technological culture and style of thinking. From the point of view of teaching methodology and psychology, the combination of fundamental scientific and engineering-practical training of specialists is a rather complex pedagogical problem;
- There is no *semantic compatibility* between different academic disciplines, which leads to "mosaic" perception of information;
- There is no systematic approach to the training of young professionals in the field of *Artificial Intelligence*;
- There is no personalization of learning, as well as an attitude to the identification, discovery and development of individual abilities;
- There is no purposeful formation of motivation for creativity;
- No formation of skills to work in real teams of developers. There is no adaptation to the real practical work;
- Any modern technology (including *Artificial Intelligence* technology) must have a high rate of development, because without it it is impossible to maintain a high level of its competitiveness. But a rapidly developing technology requires:
  - not just highly qualified personnel using and developing the technology;

- but also a high rate of improvement of this qualification, because without this it is impossible to effectively use and develop rapidly changing technology.

It follows that *learning activity in Artificial Intelligence field* and its corresponding technology should not just be an important part of *activities* in the field of *Artificial Intelligence*, but a part that is deeply integrated into all other *types of activities* in the field of *Artificial Intelligence*. Thus, for example, every *intelligent computer system* must be oriented not only to serving its end users, not only to organizing purposeful interaction with its developers who are constantly improving the system, and not only to providing a minimum "threshold of entry" for new end users and developers, but also to organize continuous and personalized professional development for each of its end users and developers in the face of constant changes made to mentioned *intelligent computer system*. To do this, the operated *intelligent computer system* must "know" what has changed in it, what it is capable of, and how to initiate these abilities (the content and form, of the corresponding user commands).

When we talk about *convergence* and *integration* in the field of *Artificial Intelligence*, we are talking not only about convergence between *intelligent computer systems*, but also between different types and fields of *human activities*. Thus, *learning activities* aimed at training specialists in *Artificial Intelligence* are organically part of *activities in Artificial Intelligence field*, and the most important way to increase the efficiency of this activity is its *convergence* and *integration* with other types of *activities in the field of Artificial Intelligence*.

7) Current state and current problems of ***organizational activities in the field of Artificial Intelligence****:* The urgent need to significantly increase the level of automation in various fields of *human activities* (industry, medicine, transportation, education, construction and many others), as well as the current results in the development of *Artificial Intelligence technology* have led to a significant increase in the creation of *applied intelligent computer systems* and the appearance of a large number of commercial organizations focused on the development of such applications. However:

- It is not easy to balance the tactical and strategic directions of development of all types of activities in the field of *Artificial Intelligence* (research activities, development of design technology and production of intelligent computer systems, development of application systems, educational activities), as well as the balance between all the above *types of activities*;
- Currently, there is no deep *convergence* of different *types of ctivities* in the field of *Artificial Intelligence* (primarily, the convergence of development of *Artificial Intelligence* technologies and development of various applied *intelligent computer systems*), which makes the development of each of these activities very difficult and in particular makes the integration of different problem-solving models (e.g., logic models, neural network models, natural language text processing models, signal processing models – audio signals, images).
- The high level of science-intensive work in the field of *Artificial Intelligence* makes special requiremnts on the qualifications of employees and their ability to work as part of *creative teams*.

D. Key tasks and methodological problems of the current stage of development of ***Artificial intelligence***

Among the **key tasks** of the current stage of development of *Artificial intelligence* should be included:

- Construction of a ***general formal theory of intelligent computer systems***, which would provide compatibility of all directions of *Artificial intelligence*, all models of knowledge representation, all models of problem solving, all components of *intelligent computer systems*. This implies:
  - Clarification of the requirements for a *new-generation intelligent computer systems* – clarifying the properties of *intelligent computer systems* that determine a high level of *intelligence*;
  - ***Convergence*** and *integration* of all kinds of *knowledge* and all kinds of *problem-solving models* within each *intelligent computer system*.
- Creating an ***infrastructure*** that ensures intensive permanent development of the *General Formal Theory of Intelligent Computer Systems* in a variety of directions, guaranteeing the preservation of logical and semantic integrity of this *theory* and compatibility of all directions of its development;
- Based on the *General Formal Theory of Intelligent Computer Systems*, constructing the ***Technology of Integrated Life Cycle Support for Next Generation Intelligent Computer Systems*** with a High Level of Interoperability and Compatibility;
- Creation of ***infrastructure*** to ensure intensive permanent development of the *Integrated technology for the development and operation of new-generation intelligent computer systems* in a variety of directions, guaranteeing the preservation of the integrity of this *technology* and compatibility of all directions of its development;
- Development of ***new-generation computers*** focused on high-performance interpretation of *logical-semantic models of next-generation intelligent computer system*;
- Creation of a ***global ecosystem of new-generation intelligent computer systems***, focused on comprehensive automation of various human activities.

The epicenter of modern methodological problems of development of *human activity* in the field of *Artificial Intelligence* is the ***convergence*** and *deep integration* of all types, directions and results of this *activity*. The level of interconnection, interaction and ***convergence*** between different types and directions of activities in the field of *Artificial Intelligence* is currently clearly insufficient. This leads to the fact that each of them develops in isolation, independent of the others. It is a question of ***convergence*** between such directions of *Artificial Intelligence* as knowledge representation, solution of intelligent problems, intelligent behavior, understanding etc., and between such types of *human activity in the field of Artificial Intelligence* as scientific research, technologies development, applications development, education, business. Why the market of *intelligent computer systems* and complex technology of *Artificial Intelligence*, providing the development of a wide range of *intelligent computer systems* for various purposes and accessible to a wide contingent of engineers, has not yet been created on the background of already long intensive development of scientific research in the field of *Artificial Intelligence*. Because the combination of high level of science intensity and pragmatism of this problem requires for its solution a fundamentally new approach to the organization of interaction between the scientists working in the field of *Artificial Intelligence*, developers of design automation tools of *intelligent computer systems*, developers of means for the realization of *intelligent computer systems*, including hardware support tools of *intelligent computer systems*, developers of applied *intelligent computer systems*. This purposeful interaction should be carried out within each of these forms of activity in the field of *Artificial Intelligence*, as well as between them. Thus, the ***convergence*** of both different types (forms and directions) of *human activities* in the field of *Artificial Intelligence* and different products (outcomes) of these activities is the main tendency of further development of theoretical and practical works in the field of *Artificial Intelligence*. It is necessary to eliminate the barriers between different types and products of activities in the field of Artificial Intelligence in order to ensure their compatibility and integrability.

Convergence of *intelligent computer systems* under development transforms a set of individual (autonomous) *intelligent computer systems* of different purposes into a collective of actively interacting *intelligent computer systems* for joint (collective) solution of complex (complex) problems and for the permanent support of compatibility between all the *intelligent computer systems* included in the collective, in the process of individual evolution of each of these systems.

The ***convergence*** of specific artificial entities (e.g., technical systems) is an aspiration to their unification (in particular, to standardization), i.e., an aspiration to minimize the diversity of forms of solving similar practical problems - an aspiration to ensure that everything that can be done equally, is done equally, but without compromising the required quality. The latter is very important, since illiterate standardization can lead to a significant brake on progress. Limiting the diversity of forms should not lead to a limitation of content, opportunities. Figuratively speaking, "words should be crowded, but thoughts - free" .

Methodologically ***convergence*** of artificially created entities (artifacts) is reduced (1) to revealing (discovering) principal similarities between these entities, which are often quite camouflaged and difficult to "see" and (2) to implementing the discovered similarities in the same way (in the same form, in the same "syntax"). Figuratively speaking, from "semantic" (semantic) equivalence we have to go to "syntactic" equivalence as well. By the way, this is exactly the point of *semantic representation of information*, the aim of which is to create such a linguistic environment (semantic space) within the limits of which (1) semantically equivalent information constructions would completely coincide, and (2) ***convergence*** of information constructions would be reduced to revealing isomorphic fragments of these constructions.

Among the general methodological problems of the current stage of development of *Artificial Intelligence* are:

- Lack of mass awareness that the creation of a market of *new-generation intelligent computer systems*, with *semantic compatibility* and a high level of *interoperability*, as well as the creation of complexes (ecosystems) consisting of such *intelligent computer systems* and providing automation of *various human activities*, is impossible unless the development teams of such systems and complexes significantly increase the level of *socialization* of **all** their employees. The level of quality of the team of developers, i.e. the level of qualification of employees and the level of coordination of their activities, should exceed the level of quality of the systems developed by this team. The considered problem of specialists' activity coherence in the field of *Artificial Intelligence* has a special meaning for the construction of *General formal theory of new-generation intelligent computer systems*, as well as the *Complex technology of development and Exploitation of new-generation intelligent computer systems*;
- Not all scientists working in the field of *Artificial Intelligence* accept the pragmatic, practical orientation of *Artificial Intelligence*;
- Not everyone accepts the need to ***converge*** the various directions of *Artificial Intelligence* and the need to integrate them in order to build a *general formal theory of intelligent computer systems*;
- Not everyone accepts the need for the ***convergence*** of different activities in the field of *Artificial Intelli-*

gence;

- An important obstacle to the ***convergence*** of scientific and technological results is the emphasis formed in science and technology on identifying differences rather than similarities. To be convinced of this, it is enough to pay attention to the fact that the level of scientific results is evaluated by scientific novelty, which can be imitated by novelty not in substance, but in form of presentation (for example, by means of new concepts or even new terms). Results in engineering, for example, in patents are also evaluated by differences from previous technical solutions. But ***convergence*** requires a different emphasis – not the search for differences, but the identification of non-obvious similarities and their transformation into obvious similarities presented in the same form;
- There is no movement to build a *comprehensive technology for the design, implementation, maintenance, re-engineering and operation of intelligent computer systems*. It is a comprehensive approach to the technological support of all stages of the life cycle of *intelligent computer systems*;
- There is no active development of work on the creation of a *global ecosystem of next-generation intelligent computer system*;
- At the heart of the modern organization and automation of *human activity* is the "*Babylonian Pillar*" of an ever-expanding variety of *languages*. This refers not only to *natural languages*, but also to *formal languages* aimed at precise representation of various kinds of *knowledge*. The variety of different *specialized languages* permeates all *human activities* – in many fields of *human activity*, *specialized languages* are created to solve different kinds of *problems*, to develop different *models for solving problems*. An example of this is the diversity of *programming languages*. *Specialized languages* can and must appear, but only as *sub-languages* of more general *languages*, the syntax of each of which coincides with the *syntax* of all the corresponding *sub-languages*. In this case within the framework of *General Formal Theory of Intelligent Computer Systems* one *universal formal language* – kernel language, with respect to which all other used *formal languages* are *sublanguages*, should be defined. *Denotational semantics* of the mentioned *universal formal language* should be set by an appropriate *formal ontology* of the highest possible level. Otherwise what ***convergence*** and *integration* of *knowledge* and *semantic compatibility* of computer systems can be talked about.

The proposed organization of *human activity* in the field of *Artificial Intelligence* is based on the following provisions:

- ***complex convergence*** - both "vertical" *convergence* between different *types of activities* in the field of *Artificial Intelligence* and "horizontal" *convergence* within each of these *activities*, corresponding to different components or different classes of *intelligent computer systems* - knowledge bases, problem solvers, different types of problem-solving models, different types of interfaces (visual, audio, natural-language), robotic intelligent computer systems, intelligent learning systems, intelligent automated control systems, intelligent design automation systems, etc.);
- ***"horizontal" convergence*** within each *human activity* in the field of *Artificial Intelligence* includes:
  - *convergence* in the *research activities in the field of Artificial Intelligence*, which means the transition from the independent development of different directions of *Artificial Intelligence* to the general theory of *intelligent computer systems*;
  - *convergence* in the development of *Artificial Intelligence technologies*, meaning the transition from the independent development of private technologies to the creation of a single set of semantically compatible private technologies;
  - *convergence* within *Artificial Intelligence engineering*, meaning the transition from the practice of independent development of various applied *intelligent computer systems* to the development of a set (ecosystem) of interoperable *intelligent computer systems*;
  - *convergence* in the framework of *educational activities in the field of Artificial Intelligence*, denoting the transition from the study of individual disciplines to the formation of young professionals a comprehensive picture of the current state of *Artificial Intelligence* and the problem directions for further development;
  - *convergence* within the *general organizational activities in the field of Artificial Intelligence*, the transition from the individual activities listed above in the field of *Artificial Intelligence* to a single set of all these activities and providing convergence and integration of these activities in the field of *Artificial Intelligence*, which will significantly improve their quality, as each of these activities is highly dependent on all others;
- organization of the design and permanent development of the proposed *technology* in the form of an ***open international project*** that provides:
  - free access to the use of the current version of the *technology* under development;
  - the opportunity for everyone to join the team of developers of this *technology*;
- ***phased*** process of forming the market of *semantically compatible* and *actively interacting* with each

other *next-generation intelligent computer systems*, the initial stages of which are:

- development of *logical-semantic models* (knowledge bases) of several *applied next-generation intelligent computer systems*;
- software implementation on modern computers *platform interpretation of logical-semantic models of next-generation intelligent computer systems*;
- installation of each developed *logical-semantic model of the applied intelligent computer system* on the developed software platform of interpretation of such models with subsequent *testing* and *re-engineering* of each such model;
- development and permanent improvement of the logical-semantic model (knowledge base) of *intelligent computer metasystem*, which contains (1) a description of the *standard of new-generation intelligent computer systems* [12], (2) a *library* of reusable (in different *intelligent computer systems*) knowledge of different types and, in particular, different *methods of solving problems*, (3) *design methods* and *design support tools* of *different types of components of intelligent computer systems* (components of knowledge bases),
- development of an *associative semantic computer* as a hardware implementation *platform for interpreting logical-semantic models of next-generation intelligent computer systems*;
- transfer of the developed *logical-semantic models of next-generation intelligent computer systems* to new, more effective variants of the implementation platform of interpretation of these models;
- development of a *new-generation intelligent computer systems market* in the form of a global ecosystem consisting of actively interacting such systems and focused on comprehensive automation of all *human activities*;
- creation of a ***knowledge market*** based on a *global ecosystem* of *next-generation intelligent computer systems*;
- automating the *re-engineering* of operating *next-generation intelligent computer systems* in the direction of bringing them into compliance with new versions of the *intelligent computer systems standard* by automatically replacing obsolete *components* in these systems with current versions of these .

It should be emphasized that the **key factor in solving the considered methodological problems** in the field of *Artificial Intelligence* are various directions of ***convergence*** and ***integration***, providing the transition to a ***new-generation intelligent computer systems***, the corresponding technology for the integrated support of their life cycle and a significant increase in the level of automation of the entire complex of human activities:

- *convergence* and *integration* of different models of *information* representation and processing in *new-generation intelligent computer systems*
  - *convergence* and *integration* of different *types* of *knowledge* in *knowledge bases* of *new-generation intelligent computer systems*
  - *convergence* and *integration* of different *problem-solving models*
  - *convergence* and *integration* of different *types of interfaces* of *new-generation intelligent computer systems*
- *convergence* and *integration* of different directions of *Artificial Intelligence* for the purpose of building a *general formal theory of new-generation intelligent computer systems*
- *convergence* and *integration* of *design* technologies for various *components of next-generation intelligent computer systems* in order to build integrated *Design Technologies for next-generation intelligent computer systems*
- *convergence* and *integration* of technologies to support various *stages of the life cycle* of *next-generation intelligent computer systems* in order to build *technologies of comprehensive support for all stages of the life cycle of next-generation intelligent computer systems*
- *convergence* and *integration* of various *types of human activities in the field of Artificial intelligence* (*research activities*, *development of technological complex*, *applied engineering*, *educational activities*) to increase the level of coherence and coordination of these *activities*, as well as to increase the level of their complex automation with the help of ***semantically compatible*** *new-generation intelligent computer systems*
- *convergence* and *integration* of various *types and fields of human activity*, as well as means of complex automation of this activity with the help of *new-generation intelligent computer systems*

The final practical result of human activity in the field of Artificial Intelligence is:

- Reorganization and complex automation of *human activity in the field of Artificial intelligence* with the help of *new-generation intelligent computer systems*;
- Step-by-step creation of a global network of effectively interacting ***new-generation intelligent computer systems***, providing comprehensive automation of various types and fields of *human activity*.

The transition from modern intelligent computer systems to *new-generation intelligent computer systems* and to the corresponding integrated technology does not require specialists in the field of Artificial intelligence to change the scope of their scientific interests. They are only required to overcome the ***Babel*** syndrome, formalizing their scientific results as part of a common collective product.

The problems of the current stage of *Artificial Intelligence* development aimed at creating a general theory and technology of *new-generation intelligent computer systems* require a fundamental integrated interdisciplinary approach and a fundamentally new organization of scientific and technical activities.

E. Comprehensive automation of human activities in the field of Artificial Intelligence with the help of new-generation intelligent computer systems

Within the framework of *OSTIS technology*, the life cycle support of new-generation intelligent computer systems (*ostis-systems*) is carried out on the basis of *OSTIS Metasystem*, which belongs to the class of *ostis-systems* and is actually a form of implementation of the specified Technology. Automation of life cycle support of *ostis-systems* is carried out both in the form of instrumental maintenance of engineering activities (in particular, OSTIS Metasystem is a system of design automation of ostis-systems), and in the form of information maintenance of the specified activities. For this purpose, the knowledge base *OSTIS Metasystem* contains:

- The current state of the full text *Standard of ostis-systems*;
- The current state of the reusable components library of ostis-systems;
- The current status of the ostis-systems life cycle support methodologies used and implemented by engineers;
- Documentation of tools used by engineers to support the life cycle of ostis-systems.

In addition to all of this, *OSTIS Metasystem*:

- Provides automation of *the support of the life cycle of the ostis-systems Standard*, i.e. provides the organization of interaction between the authors of this Standard, aimed at its permanent development
- Provides automation of *OSTIS Technology life cycle Support*, which boils down to supporting the life cycle of the main part of the *OSTIS Metasystem* knowledge base, which is the complete documentation of the current state of *OSTIS Technology*.

Automation of other directions of *human activities in the field of Artificial Intelligence* can also be done with *ostis-systems* that are semantically compatible and interact with *Metasystem OSTIS* within the *Eco-system OSTIS*.

IV. Problems and prospects of comprehensive automation of all types and directions of human activity with the help of next-generation intelligent computer systems

Above it was considered how the whole complex of *Human activity in the field of Artificial Intelligence* is carried out and automated with the help of new-generation intelligent computer systems. Now we will summarize it and consider the principles of organization and complex automation of *human activity* as a whole, i.e. automation of the most various types and field of human activity.

A. General principles of systematization of human activity and its comprehensive automation with the help of new-generation intelligent computer systems

The experience of complex organization, structuring and automation of *human activities in the field of Artificial Intelligence* (in the creation and maintenance of intelligent computer systems) can be generalized to other fields of human activities. This is due to the following reasons:

- Firstly, because human activity aimed at supporting the whole *life cycle of new- generation intelligent computer systems* is a paricular direction of activity in relation to the type of human activity aimed at supporting the whole life cycle of any artificial (artificially created) entity (any artifact). Depending on the complexity of the artificially created entity, the level of complexity of human activities aimed at supporting the life cycle of this entity can be very different. But the overall structure of these activities corresponding to the different stages of the life cycle of artificially created entities. As well as the necessary directions of providing this engineering activity is the same for artificial entities of different classes. These directions of providing support for the life cycle of artificial entities include:
  - research activities aimed at the study of artificial entities of the relevant class;
  - development of the standard of artificial entities of the specified class;
  - development of life-cycle support technology for the specified class of artificial entities;
  - training personnel capable of supporting the life cycle of the specified class of artificial entities, i.e., capable of effectively using the above-mentioned technology;
  - training personnel capable of participating in the above-mentioned research and development activities;
  - training personnel capable of participating in the development of the standard of artificial entities of a given class;
  - training personnel capable of participating in the design and development of the above-mentioned technology;
  - organizational support of the whole set of works on the development and use of the above-mentioned technology.
- Second, because many complex technical systems are actually becoming *intelligent computer systems* (including distributed ones) with various sets of sensor and effector subsystems – intelligent cars

with autopilot and autosteer, intelligent automatic factories, smart houses, smart cities, etc.

- Thirdly, because the nature of the activities of *new-generation intelligent computer systems* and the nature of each *person* and each organization in fact there is little difference, because the *new-generation intelligent computer systems* become equal partners (subjects) of human activity, because the level of their independence, responsibility, interoperability and intelligence is close to the corresponding qualities of the *natural* subjects of human activity (individuals, legal persons, legal entities, etc.)

So, the structuring of *human activity* in the field of *Artificial Intelligence* based on the concepts of *type of activity*, *field of activity*, *product of activity* (object of activity) can be easily generalized for all *scientific and technical disciplines*, which makes it possible to consider automation of activity within all *scientific and technical disciplines* from the general position, as. because automation of different *type of activity* within different *scientific and technical disciplines* can look similar, and sometimes can be implemented using the same *intelligent computer system*. So, for example, any *intelligent computer system for design automation* of technical systems of a given types can be built on the basis of *intelligent computer system for design automation and knowledge base reengineering*, since the result of design of any *technical system* is a formal model (description, specification, documentation) of this *technical system*, which has enough completeness to reproduce (implement) this system.

At the current stage of development of *Artificial Intelligence* it is necessary to move from automation of separate *types of human activity* to integrated automation of the whole complex of *human activity*, to creation and constant evolution of the whole ***global ecosystem of intelligent computer systems***. That systems independently interact both among themselves and with people whose activities they automate, and also with modern computer systems, which are not intelligent systems. It should be remembered that the main "overheads", the main problems, arise at the "joints" when integrating different technical solutions. The developer of each subsystem should guarantee the absence of such "overhead" costs. It should be emphasized that one should focus not so much on creating an effective *global ecosystem of intelligent computer systems* as on creating effective techniques and tools aimed at the *permanent evolution* of such an *ecosystem*.

The methodology of complex automation *human activity* includes the following steps:

- Construction of a general ***structure of human activity***, based on the hierarchy of *human activity* by types of activity and products of activity with a clear fixation of different kinds of connections between the various components of this structure.
- Formalization of various *types of human activity*.
- Development of ***technology***, which ensures the maximum possible automation of this activity with the help of *new-generation intelligent computer systems*.
- Ensuring maximum possible ***convergence*** of various *types of activities*, which will reduce the variety of automation tools (i.e., appropriate *new-generation intelligent computer systems*).
- Ensuring maximum possible ***convergence*** of technologies for performing the same *type of activity* for different objects of activity (convergence of design technologies for objects of different classes, convergence of monitoring, prevention and diagnosis technologies for agents of different classes, etc.) and thereby ensure ***convergence*** of corresponding automation tools based on *new-generation intelligent computer systems*

B. Multiplicity of types of human activities and the connections between them

The basic type of human activity can be considered ***supporting the life cycle*** of various entities.

The class of objects of activity for this *type of activity* is the class of all kinds of socially significant objects, which it makes sense to influence, support the life cycle of which is advisable to implement.

life cycle support
:= [support of the life cycle of socially significant entities]
∈ *type of activity*
⇒ *particular type of activity performed at some point**:
- *design*
- *production*
- *initial training*
  := [setting up]
- *quality monitoring*
  := [scheduled examination and diagnosis]
- *restoring the required level of quality*
  := [repair, treatment]
- *reengineering*
  := [renewal, improvement]
- *security*
- *using*
  := [operation, usage]

⇒ *particular type of activity over a subclass of activity objects**:
- *research activities*
  := [support for the life cycle of scientific theories]
  ⇒ *class of activity objects**:
  scientific theory
- *standardization*
  := [standards life cycle support]
  ⇒ *class of activity objects**:
  standard
- *support for the technology life cycle*
  ⇒ *class of activities object**:
  technology
- *educational activities*

:= [support for the human resource life cycle]
⇒ *class of activity objects**:
resource personnel

- *supporting the life cycle of comprehensive support management metasystems and providing support for the life cycle of the entities of the respective classes*
  ⇒ *class of activity objects**:
  meta-system for the integrated management of the support and provisioning of the life cycle of the entities of the respective classes

When the general structure of *human activity* was considered above by summarizing the structure of ***human activity in Artificial Intelligence***, we:

- introduced the concept of *type of activity*
- As a "starting point" generalization we chose such a *type of activity* as *support of life cycle of intelligent computer systems*
- further expand the class of *activity objects* of the selected *type of activity*,
  - moving from the class of *intelligent computer systems* to the class of all kinds of *artificial material entities*
  - combining the class of *artificial material entities* with the class of *natural material entities* (material entities of natural origin), as well as with the class of *natural-artificial material entities* (either *natural artificially modified material entities* or *hybrid natural-artificial material entities* with both natural and artificial components);
  - combining the class of *material entities* with the class of *information resources*, i.e., socially significant information constructions (documents) which are the products of corresponding actions or activities (*scientific theories*, *standards*, *bases of knowledge*, *methods*, *design documents* of corresponding created objects)
  - combining the class *material entities information resources* with the class *material-information-objects*, which, in particular, include various technologies.

Thus, *support of the life cycle* of various socially important objects is a special type of *human activity*. Firstly, the efficiency of *human activity* in general depends (1) on the duration of the socially useful (active) phase of the life cycle of the used objects and (2) on the amount of society's expenditure on maintaining the necessary socially useful properties of the used objects. Secondly, the nature and *technology* of life cycle support of different types of socially important objects can differ significantly from each other. For example, the organization of life cycle support for automobiles, traditional computer systems of various purposes, modern intelligent computer systems, interoperable intelligent computer systems, people, enterprises, houses, various legal entities, settlements, etc. differs significantly. At the same time, the typology of socially significant objects whose life cycle must be supported includes the most diverse classes of objects - artificially created material information products of human activity, all people, all kinds of social communities and enterprises. The diversity of types of socially significant objects generates a variety of technologies corresponding to them, which complicates the complex automation of human activity as a whole.

Nevertheless, we note that *types of human activity* is much smaller than *fields of human activity*. This is, to some extent, due to the fact that the types of relations between entities (relative concepts) are much fewer than the classes of various entities. This circumstance indicates that the movement in the direction of global automation of *society* activities should be based on the orientation towards a competent systematization of *types of human activities*, and their deepest *convergence* (both within each type of activity and between different types). Thanks to this, the artificially introduced variety of automation means of *human activities* can be minimized.

should be distinguished*
∋ {● *research activities*
:= [support for the life cycle of scientific theories]
● *standardization*
:= [standards design and development]
:= [standards life cycle support]
● *support for the technology life cycle*
}

Research activity is aimed at studying the entities of a given class, at studying the principles underlying their structure and functioning. Within this type of activity, novelty and competition of ideas and approaches are important, the correlation between the structure (architecture) of the organization of functioning of the studied *entities* and the general characteristics (parameters) of the quality of these entities, the general requirements imposed on them is important. The product of the activity under consideration is the *General theory of entities of a given class*, which reflects the plurality and even contradiction of different points of view and whose most important direction of development (evolution) is to bring together (*convergence*) different points of view and ensure compatibility and inconsistency between them. At the heart of *research activity* is the competition of points of view, the principal novelty of ideas and verified results aimed at revealing and substantiating non-obvious properties and regularities of the corresponding *subject domain*, at developing methods of solving various *classes of problems* solved within this *subject domain*. The purpose of the *research activity* and the required detailing of the generated knowledge about the research objects of the corresponding *subject domain*

In contrast to *research activity*, the development of the *standard* of created entities and the development of the corresponding *technology* of support for their life cycle is based on agreement of different points of view (consensus search) and their simplification as much as possible (observance of Occam's Razor principle). The necessity of such a methodological attitude is caused by the mass nature of *human activity* to create and *support* the life cycle of the corresponding class of entities and the need to involve people with *different* (including quite low) qualifications in this activity. In the process of *development of the standard of entities of a given class* it is not the competition of different points of view that is important, but their *convergence*, *semantic compatibility* and deep integration. Each *standard artificial entities of a given class* is an agreed currently point of view (consensus) about the structure, functioning, properties, and patterns of artificial entities of a given class, an agreed (generally accepted) part of the *General Theory of Artificial Entities of a Given Class*, which is understandable to the broad contingent of practitioners (engineers) who design, produce and maintain the entire life cycle of specific *artificial entities of a specified class*.

The creation and *support of the technology life cycle* must take into account a number of requirements for any *technology*:

- comprehensiveness - maximum possible coverage of all tasks that must be solved with *technology* (at least all stages of the life cycle)
- maximum ease of use of *technology* (required completeness of documentation, intelligent help-support, absence of unnecessary information that is not necessary for using *technology*, availability of rich and systematic library of typical reusable solutions)

Society is a hierarchical system of interacting individual and collective entities, each of which:

- Produces either part of the socially significant products produced by the collective entity, which includes this subject, or an integral socially significant product (produced goods) consumed by other external entities or provides some service to another entity, aimed at ensuring the livelihood and improvement of this other entity.
- It consumes products produced by other entities, necessary for the production of its own products (raw materials and equipment), as well as necessary to ensure its own livelihood.
- It consumes services provided by other entities necessary for the production of its own products or services, as well as those necessary to improve its activities.

The main directions of automation of the whole complex *human activity* are:

- automation of socially useful professional activities of all subjects of activity (both individual subjects - all individuals, and all kinds of collective - corporate entities, including legal entities)
- automation of providing (creating) comfortable conditions for all subjects of society based on the monitoring of activities and specific (adapted) facilitation of the evolution of each subject, taking into account its immediate needs and problems.

The organization of each subject's interactions with the external environment should be carried out both by this subject and by the mentioned external environment (i.e., by society). Society should turn "face" to each subject and not throw it to the mercy of fate. At present, creation (provision) of conditions of society's subjects' activity is given to each such subject. Society, represented by other subjects designated for this purpose, provides services and supplies goods on the initiative of the subject in need. Thus, the responsibility for the development of each subject of activity falls exclusively on the "shoulders" of this subject. The society's support is general and does not take into account in any way the peculiarities of the current situation of each subject.

The most important reason preventing further increase in the overall level of automation of human activity is the fact that automation of various fields of human activity is carried out local. At the current stage of application of intelligent computer systems the main problem is not the automation of local types and fields of human activity, but the automation of complex processes of human activity, requiring *integration* in a priori unpredictable combinations of a variety of information resources and a variety of automated services, implemented in the form of specialized intelligent computer systems.

Locality of automation of human activity leads to the fact that all human activity acquires the appearance of "archipelago" consisting of well-automated "islands" but interconnected "manually". This "manual" non-automated connection of these "islands" depends entirely on the human factor and qualification of corresponding executors.

The specified "manual" connection of some set of semantically close automated fields of human activity can be automated, but it should be done very competently at a high level of system culture and on the fundamental basis of the general theory of human activity.

Another important reason preventing further improvement of the overall level of automation in society is that automation of different fields of human activity is carried out without identifying and deeply analyzing the similarities of some activities in different fields and, accordingly, without converging, ***convergence*** and ***unifying*** these *types of activity*.

The most important direction to increase the level of automation of human activity is the transition to automation of more and more complex (large) types and fields of human activity, for example, from the automation of various enterprises, organizations, economic services to the automation of the city as a whole).

Automation of complex human activities requires the creation of a set of actively interacting computer systems, each of which provides automation of the corresponding particular type of human activity that is part of the complex activity being automated. In this case the number of levels of hierarchy of automated human activities is not limited in any way. Obviously, the level of automation of complex human activities is determined by:

- level of convergence (convergence, compatibility) of the respective particular types of activities;
- quality of integration of these private activities;
- level of convergence of computer systems, providing automation of the specified particular types of activities;
- quality of interaction of these computer systems, i.e. the level of interoperability of these systems).

The level of evolution of society depends to a large extent on the level of automation of human activity, on the level of development of the corresponding technologies of such automation. But this dependence looks much more complicated than it seems at first glance, especially if we are talking about automation of not physical but intelligent human activity (both individual and collective). Illiterate and, all the more so, socially irresponsible or malicious automation of society's informational activity can cause enormous damage to its development. Such illiteracy and irresponsibility, for example, leads to such side factors as computer addiction, virtualization of the environment, superficiality of thinking, reduction of cognitive motivation and activity, and much more. Consequently:

- Needs to significantly increase the level of social responsibility of the developers of computer systems and related technologies.
- The danger from illiterate, socially irresponsible and, even more so, malicious implementation of the new-generation of intelligent computer systems can be fatal for humanity.

If we consider *society* as a *multi-agent system* consisting of independent intelligent agents, it is obvious that the most important factors determining the improvement of the quality (level of development) of *society* are:

- increase the efficiency of humanity's *knowledge* and *skills* to use the experience accumulated by *society*, the efficiency of humanity's *knowledge* and *skills*;
- increase the rate of acquisition, accumulation and systematization of humanity's effective use of *knowledge* and *skills*.

The solution of the above problems becomes quite possible by using *new-generation intelligent computer systems*, by means of which humanity's accumulated the *knowledge* and *skills* will be organized as a systematized distributed library of reusable information resources (*knowledge* and *skills*). Consequently, the systematization and automation of the reusable information resources(that was acumaleted by humans) requires their convergence, deep integration and formalization. A special place in this process is occupied by mathematics as a basis for systematization and formalization of knowledge and skills at the level of formal *ontologies of the upper level*.

V. Conclusion

Due to the fact that *new-generation intelligent computer systems* become independent and active subjects of *human activity* sufficiently equal to humans(natural individual subjects of human activity), the nature and, respectively, the level of automation of *human activity* changes significantly – the need to control automation means is removed, since this "manual" management is replaced by the distribution of duties and responsibilities among people

If automation of any kind of automation in any types of *human activity* is carried out with the help of *new-generation intelligent computer systems* and if *new-generation intelligent computer systems* that provide automation of different types and fields of *human activity* will meaningfully interact with each other, the overall level of automation of *human activities* will significantly increase due to the fact that there will be no need to manually coordinate the use of various automation tools.

The efficiency and labor intensity of automation of different types and fields of *human activities* will be significantly determined by the degree of ***convergence*** between different types and fields of *human activities*. A hierarchical model of *human activity* must be built, within which a competent systematization and stratification of all types and fields of *human activity* must be carried out, aiming against an excessive eclectic diversity. Thus, before implementing comprehensive automation of *human activities* with the help of *new-generation intelligent computer systems*, it is necessary to rethink the organization of this activity from the perspective of general systems theory. Otherwise, automating clutter will lead to more clutter.

Let us emphasize the fact that many of the problems we have considered the current state and directions of further development of *human activity in the field of Artificial Intelligence* are similar to the problems and trends in many other scientific and technical disciplines.

Each person's time is the main irreplaceable resource of society, and it should be spent not on the routine support of the life cycle of all kinds of socially important objects, but on the integrated development of appropriate *technologies*. Automation of human activities with the help of a global system of interoperable semantically compatible and actively interacting *intelligent computer systems* in various fields of *human activities* will significantly reduce the time of each person to perform routine, easily automated activities. Human activity should become oriented to the maximum possible self-realization, opening creative potential of each person, aimed at accelerating the rate of increasing the level of intelligence of the whole society.

The creation of *a Global ecosystem of next-generation intelligent computer systems*, involves:

- Building a formal model of *human activity*;
- The transition from eclectic construction of complex *intelligent computer systems* that use different types of *knowledge* and different types of *problem-solving models* to their deep *integration* and *unification*, when the same representation models and knowledge processing models are implemented equally in different systems and subsystems;
- Reducing the distance between the modern level of *the theory of intelligent computer systems* and the practice of their development;
- The development of a competent tactic and strategy for the transition period, in which modern *intelligent computer systems* should be gradually replaced by *new-generation intelligent computer systems*, which should effectively interact not only with each other, but also with well-proven modern information resources and services.

Acknowledgment

The authors would like to thank the scientific teams of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics and the Brest State Technical University for their assistance and valuable comments.

References

[1] K. Yaghoobirafi and A. Farahani, “An approach for semantic interoperability in autonomic distributed intelligent systems,” *Journal of Software: Evolution and Process*, vol. 34, no. 10, p. e2436, 2022.

[2] Ouksel, A. M. and Sheth, A., “Semantic interoperability in global information systems,” *SIGMOD Rec.*, vol. 28, no. 1, p. 5–12, mar 1999.

[3] Lanzenberger, Monika and Sampson, Jennifer and Kargl, Horst and Wimmer, Manuel and Conroy, Colm and O’Sullivan, Declan and Lewis, David and Brennan, Rob and Ramos-Gargantilla, José Ángel and Gómez-Pérez, Asunción and Fürst, Frédéric and Trichet, Francky and Euzenat, Jérôme and Polleres, Axel and Scharffe, François and Kotis, Konstantinos, “Making ontologies talk: Knowledge interoperability in the semantic web,” *IEEE Intelligent Systems*, vol. 23, no. 6, pp. 72–85, 2008.

[4] Francila Weidt Neiva and José Maria N. David and Regina Braga and Fernanda Campos, “Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature,” *Information and Software Technology*, vol. 72, pp. 137–150, 2016.

[5] J. Pohl, “Interoperability and the need for intelligent software: A historical perspective,” 09 2004.

[6] Jeff Waters and Brenda J. Powers and Marion G. Ceruti, “Global interoperability using semantics, standards, science and technology (gis3t),” *Computer Standards & Interfaces*, vol. 31, no. 6, pp. 1158–1166, 2009.

[7] V. Golenkov, N. Guliakina, I. Davydenko, and A. Eremeev, “Methods and tools for ensuring compatibility of computer systems,” in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed. BSUIR, Minsk, 2019, pp. 25–52.

[8] A. Palagin, “Problemy transdisciplinarnosti i rol’ informatiki [problems of transdisciplinarity and the role of informatics],” *Kibernetika i sistemnyj analiz [Cybernetics and Systems Analysis]*, no. 5, p. 3–13, 2013.

[9] V. Tarasov, *Ot mnogoagentnykh sistem k intellektual’nym organizatsiyam [From multi-agent systems to intelligent organizations]*. M.: Editorial URSS, 2002, (in Russian).

[10] I. Barinov, , N. Borgest, S. Borovik, O. Granichin, S. Grachev, Y. Gromyko, R. Doronin, S. Zinchenko, A. Ivanov, V. Kizeev, R. Kutlakhmetov, V. Laryukhin, S. Levashkin, A. Mochalkin, M. Panteleev, S. Popov, E. Sevastyanov, P. Skobelev, A. Chernyavsky, V. Shishkin, and S. Shlyaev, “Development strategy formation of the committee on artificial intelligence in the scientific and educational center "engineering of the future",” *Ontology of Designing*, vol. 11, no. 3, pp. 260–293, Sep. 2021. [Online]. Available: https://doi.org/10.18287/2223-9537-2021-11-3-260-293

[11] A. Iliadis, “The tower of babel problem: Making data make sense with basic formal ontology,” 02 2019.

[12] V. Golenkov, N. Guliakina, and D. Shunkevich, *Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

Проблемы и перспективы автоматизации различных видов и областей человеческой деятельности с помощью интеллектуальных компьютерных систем нового поколения

Голенков В. В., Таранчук В. Б., Ковалёв М. В.

В работе рассмотрены принципы автоматизации различных областей человеческой деятельности с использованием интеллектуальных компьютерных систем нового поколения. Предлагается онтология различных видов деятельности и соответствующих технологий. Детализация указанных принципов осуществляется на примере человеческой деятельности в области Искусственного интеллекта.

Received 01.11.2022

Principles for implementing the ecosystem of next-generation intelligent computer systems

Alexandr Zagorskiy
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: alexandr.zagorskiy.research@gmail.com

***Abstract*—In the article, the architecture of the ecosystem of intelligent computer systems based on the OSTIS Technology is considered. The formal interpretation of such concepts as ostis-system, ostis-community has been clarified, the typology of ostis-systems has been identified, which together makes it possible to determine the structure of the OSTIS Ecosystem. The results obtained can be applied in the implementation of such projects as "Society 5.0", "Industry 4.0", "Smart Home", "Smart City", "Knowledge Market".**

***Keywords*—Digital Ecosystem, OSTIS Ecosystem, Society 5.0, Smart City**

I. Introduction

To increase the level of automation of more and more broad types of human activity, a qualitative transition to the development of entire complexes of independently interacting intelligent computer systems is necessary.

The central problem of the next stage in the development of information technologies is the problem of ensuring the semantic compatibility of computer systems and their components [1]. To solve this problem, it is necessary to move from traditional computer systems and modern intelligent systems to semantically compatible computer systems [2], [3].

Semantic computer systems are next-generation computer systems that eliminate many of the shortcomings of modern computer systems. *However, for the mass development of such systems, an appropriate technology is required*, which should include:

- methods and tools for designing semantic computer systems;
- methods and tools for permanent improvement of the technology itself.

As the subject of engineering activity in the field of artificial intelligence, not a set of intelligent computer systems should be considered but the whole complex of intelligent computer systems interacting with each other.

The purpose of this work is to designate the architecture of the ecosystem, within which the most comfortable conditions would be created for the implementation of next-generation intelligent computer systems, capable of organizing collectives of systems due to the high level of interoperability of these systems. It is necessary to focus not on creating an ideal information ecosystem but on creating an effective technology aimed at the permanent evolution of this ecosystem.

II. Analysis of existing approaches to solving the problem

To create any complex things, humanity draws on the concepts of nature. The process of studying elementary, natural systems and processes in order to inspire and try to replicate the learned behaviour in their designs is called biomimicry. Biomimicry is not a direct imitation of processes – it is the study of basic natural principles and their application in various areas of human society [4].

The idea of a network, a community, complex adaptive systems has also been realized by nature. The concept of such an archetype is reduced to the combination of many autonomous objects with each other. These objects are strongly connected to each other and at the same time do not have any center. Thus, they form a decentralized network, where there is the lack of a single control center [5]. The following characteristics of such a system can be distinguished:

- absence or lack of centralized control;
- autonomous nature of participants, objects of such a network;
- strong connectivity of the participants of such a network with each other;
- the influence of the participants of such a network on each other is non-linear and rather complicated.

Such distributed artificial systems have both advantages (high level of adaptability, stability, connectivity) and disadvantages (non-optimality, uncontrollability, unpredictability of behaviour). The most appropriate example of an implemented technology based on the concept of a network is the Internet.

It is convenient to use the network archetype to display complex processes, the interdependence of components, economic, social, environmental processes, and communication processes. In such processes, there is no beginning or ending, everything is the center. The network is the only topology capable of limitless expansion or self-learning;

other topologies have their own limitations. “The Atom is the icon of 20th century science. The symbol of science for the next century is the dynamical Net” [6].

The idea of a digital ecosystem is also borrowed from nature, where biological ecosystems are the main source of inspiration. The concept of an ecosystem has become a popular way of describing collaboration outside an organization [7]. It can be defined as a multi-stakeholder structure of organizations that materializes a shared value proposition. Ecosystems have two distinct characteristics compared to other collaborative concepts: complementarity and interdependence exist at the same time, and the system is not fully hierarchically controlled [8].

Implementation options for the ontology of the digital ecosystem have been proposed [9]. There are such types of ecosystems as Business, Innovation, Knowledge, Entrepreneurial. Each of these ecosystems has its own characteristics, structure, purposes [10].

The ecosystem serves as a value multiplier for a product. The coefficient value depends on the quality of each member of the given community [11], [12]. Approaches to the integration of the digital ecosystem into various spheres of human activity [13], [14], [15], [16], approaches to expanding an existing ecosystem with new objects [17] are considered. There were also attempts to recreate the ecosystem model based on traditional information technologies [18], [19].

With traditional approaches to solving the problem of ecosystem formation, there are problems associated with the low level of interoperability of such systems [20]. Often, each of the systems will have its own specialized programming interface and data format for communicating with it, which leads to additional costs for eliminating the shortcomings of such problems. Moreover, life cycle support, modification of existing systems can impose additional time and resource costs.

III. Proposed approach

Within this article, it is proposed to take an OSTIS Technology [21] as a basis. The OSTIS Technology is a set of technologies that provide the design, production, operation, and reengineering of intelligent computer systems designed to automate a wide variety of human activities. The Technology is based on semantic representation and ontological systematization of knowledge, as well as agent-oriented knowledge processing.

The principles underlying the OSTIS Technology are:

- Orientation towards the development of intelligent computer systems with a high level of intelligence and, in particular, a high level of socialization. These systems, developed using the OSTIS Technology, will be called ostis-systems.
- Orientation towards complex automation of all types and areas of human activity by creating a network of interacting and coordinating their activities ostis-systems. This network of ostis-systems, together with their users, is called the OSTIS Ecosystem.
- The OSTIS Technology is implemented as a network of ostis-systems, which is part of the OSTIS Ecosystem. The key ostis-system of this network is the OSTIS Metasystem (Intelligent MetaSystem for ostis-systems), which implements the OSTIS Technology Core, which includes basic (subject-independent) methods and tools for designing and producing ostis-systems with integration into their structure of typical built-in support subsystems for operation and reengineering of ostis-systems. The remaining ostis-systems that are part of the network under consideration implement various specialized ostis-technologies for designing various classes of ostis-systems that automate any areas and types of human activity.
- Convergence and integration based on the semantic representation of knowledge of various scientific directions of Artificial Intelligence (in particular, various basic knowledge and skills for solving intelligent problems) within the General formal semantic theory of ostis-systems.
- Orientation towards the development of next-generation computers that provide efficient (including productive) interpretation of the logical-semantic models of ostis-systems, which are represented by knowledge bases of these systems with semantic representation.

Within the technology, several universal variants of visualization of *SC-code* constructions are proposed, such as *SCg-code* (graphic variant), *SCn-code* (nonlinear hypertext variant), *SCs-code* (linear string variant).

Within this article, fragments of structured texts in the SCn code [22] will often be used, which are simultaneously fragments of the source texts of the knowledge base, understandable both to a human and to a machine. This allows making the text more structured and formalized, while maintaining its readability. The symbol “:=” in such texts indicates alternative (synonymous) names of the described entity, revealing in more detail certain of its features.

The basis of the knowledge base within the OSTIS Technology is a hierarchical system of subject domains and ontologies. Based on this, in order to solve the problems set within this article, it is proposed to develop the following system of subject domains and ontologies:

IV. Formal model of the ecosystem of next-generation intelligent computer systems

An OSTIS Ecosystem is a sociotechnical ecosystem, which is a collective of interacting semantic computer systems, which provides permanent support for the evolution and semantic compatibility of all its member systems throughout their entire life cycle.

Subject domain and ontology of the OSTIS Ecosystem

⇒ *private subject domain**:

- *Logical-semantic model of integration of heterogeneous information resources and services in the OSTIS Ecosystem in the process of its extension*
- *Subject domain and ontology of semantically compatible intelligent ostis-portals of scientific knowledge*
- *Subject domain and ontology of semantically compatible intelligent corporate ostis-systems for various purposes*
- *Subject domain and ontology of ostis-systems, which are personal assistants of users, ensuring the organization of effective interaction of each user with other ostis-systems and users that are part of the OSTIS Ecosystem*

An intelligent computer system that is built in accordance with the requirements and standards of the OSTIS Technology is defined as an ostis-system. This provides a significant development of a number of properties for this computer system, which can significantly increase the level of intelligence of this system (and, above all, its level of learning and the level of socialization).

The OSTIS Ecosystem is a collective of interacting:

- ostis-systems themselves;
- users of the specified ostis-systems (both end users and developers);
- some computer systems that are not ostis-systems but are considered by them as additional information resources or services.

Members of the OSTIS Ecosystem collective are characterized as:

- semantically compatible;
- constantly evolving individually;
- constantly maintaining their compatibility with other members in the course of their individual evolution;
- capable of decentralized coordination of their activities.

The purpose of the OSTIS Ecosystem is to provide continuous support for the compatibility of computer systems included in the Ecosystem both at the stage of their development and during their operation. The problem lies in the fact that during the operation of the systems included in the OSTIS Ecosystem, they may change, due to which compatibility may be violated. The objectives of the OSTIS Ecosystem are:

- prompt implementation of all agreed changes in the ostis-systems standard (including changes in the concepts systems used and their corresponding terms);
- permanent support of a high level of mutual understanding of all systems included in the OSTIS Ecosystem, as well as all their users;
- corporate solution for various complex problems that require the coordination of several ostis-systems and possibly also particular users.

The OSTIS Ecosystem is a transition from independent ostis-systems to collectives of independent ostis-systems, i.e. to distributed ostis-systems.

ostis-system

⇒ *subdividing**:

{● *stand-alone ostis-system*
● *built-in ostis-system*
● *ostis-systems collective*
}

A. Compatibility support between ostis-systems that are part of the OSTIS Ecosystem

Each system that is part of the OSTIS Ecosystem have to:

- study intensively, actively, and purposefully, both with the help of teachers-developers and independently;
- inform all other systems about proposed or finally approved changes in ontologies and, in particular, in the set of concepts used;
- accept proposals from other ostis-systems about changes in ontologies, including the set of concepts used, to agree or approve these proposals;
- implement approved changes to ontologies stored in its knowledge base;
- help in maintaining of a high level of semantic compatibility not only with other ostis-systems included in the OSTIS Ecosystem but also with its users (train them, inform them about changes in ontologies).

Special requirements are imposed on independent ostis-systems that are part of the OSTIS Ecosystem:

- they must have all the necessary knowledge and skills for messaging and purposeful organization of interaction with other ostis-systems that are part of the OSTIS Ecosystem;
- in the context of constant change and evolution of ostis-systems included in the OSTIS Ecosystem, each of them must itself monitor the state of its compatibility (consistency) with all other ostis-systems, i.e. must independently maintain this compatibility, coordinating with other ostis-systems all changes that require coordination, occurring in itself and in other systems.

According to the purpose, the ostis-system, included in the OSTIS Ecosystem, can be:

- assistants of specific users or specific user collectives;
- typical built-in subsystems of ostis-systems;
- systems of information and instrumental support for designing various components and various classes of ostis-systems;
- systems of information and instrumental support for designing production of various classes of technical and other artificially created systems;
- portals of knowledge in various scientific disciplines;
- control automation systems for various complex objects (manufacturing enterprises, educational institutions, university departments, certain students);
- intelligent reference and help-systems;
- intelligent robotic systems.

To ensure high operational efficiency and high rates of evolution of the OSTIS Ecosystem, it is necessary to constantly increase the level of information compatibility (the level of mutual understanding) not only between the computer systems that are part of the OSTIS Ecosystem but also between these systems and their users. One of the ways to ensure such compatibility is the desire to ensure that the knowledge base (picture of the world) of each user becomes a part (fragment) of the Unified knowledge base of the OSTIS Ecosystem. This means that each user must know how the structure of each scientific and technical discipline is arranged (objects of research, subjects of research, definitions, regularities, etc.), how different disciplines can be interconnected.

Maintaining the compatibility of the OSTIS Ecosystem with its users is carried out as follows:

- each ostis-system includes built-in ostis-systems oriented
  - for permanent monitoring of the activities of end users and developers of this ostis-system,
  - for analyzing the quality and, first of all, the correctness of this activity,
  - for advanced training of users (personalized training);
- The OSTIS Ecosystem includes ostis-systems specially designed to train users of the OSTIS Ecosystem with basic generally recognized knowledge and skills for solving the corresponding classes of problems.

The OSTIS ecosystem is associated with its unified knowledge base, which is a virtual combination of the knowledge bases of all ostis-systems that are part of the OSTIS Ecosystem. The quality of this knowledge base (completeness, consistency, compliance) is a constant concern of all independent ostis-systems that are part of the OSTIS Ecosystem.

B. Structure of the OSTIS Ecosystem

A subject that is part of the OSTIS Ecosystem is an agent of the OSTIS Ecosystem.

```
OSTIS Ecosystem agent
⇒    subdividing*:
     {•    individual ostis-system of the OSTIS
           Ecosystem
           ⇒    subdividing*:
                {•    stand-alone ostis-system of
                      the OSTIS Ecosystem
                 •    build-in ostis-system of the
                      OSTIS Ecosystem
                }
      •    ostis-community
           ⇒    subdividing*:
                {•    simple ostis-community
                 •    hierarchical
                      ostis-community
                }
      •    OSTIS Ecosystem user
     }
```

The concept of an ostis-community is not only a collective of ostis-systems but also a certain collective of humans (users and developers of the corresponding ostis-systems). The ostis-community is a stable fragment of the OSTIS Ecosystem, which provides comprehensive automation of a certain part of the collective human activity and a permanent increase in its efficiency. A hierarchical ostis-community is an ostis-community, at least one of whose members is some other ostis-community.

Rules of behaviour for OSTIS Ecosystem agents:

- Coordinate the denotational semantics of all used signs (primarily concepts);
- Coordinate terminology, eliminate contradictions and information holes;
- Constantly eliminate synonymy and homonymy both at the level of sc-elements (internal characters) and at the level of their corresponding terms, as well as other external identifiers (external designations);
- Each agent of the OSTIS Ecosystem, on its own initiative, can become a member of any ostis-community of the OSTIS Ecosystem after appropriate registration.

All rules of behaviour for OSTIS Ecosystem agents must be observed not only by ostis-systems that are agents of the OSTIS Ecosystem but also by human who are agents. The correct behaviour of ostis-systems as agents of the OSTIS Ecosystem is much easier to ensure than the correct behaviour of human as such agents. The behaviour of users (natural agents) of the OSTIS Ecosystem must be closely monitored and controlled, constantly contributing to the improvement of their qualifications as agents of the OSTIS Ecosystem, as well as increasing their level of motivation, purposefulness, and self-realization.

The OSTIS Ecosystem is the maximum hierarchical ostis-community that provides comprehensive automation of all types of human activity. It cannot be part of any other ostis-community. The principles underlying the OSTIS Ecosystem are:

- the OSTIS ecosystem is a network of ostis-communities;
- each ostis-community corresponds one-to-one with the corporate ostis-system of this ostis-community;
- each ostis-community can be a part of any other ostis-community on its own initiative. Formally, this means that the corporate ostis-system of the first ostis-community is a member of another ostis-community;
- each specialist who is part of the OSTIS Ecosystem is assigned an one-to-one correspondence with their personal ostis-assistant, which is considered as a corporate ostis-system of a degenerate ostis-community consisting of one human.

In the OSTIS Ecosystem, the following levels of hierarchy can be distinguished:

- individual computer systems (individual ostis-systems and humans who are end users of ostis-systems);
- a hierarchical system of ostis-communities, each of which can have members of individual ostis-systems, humans, and other ostis-communities;
- the maximum ostis-community of the OSTIS Ecosystem that is not a member of any other ostis-community, which is part of the OSTIS Ecosystem.

The quality of the OSTIS Ecosystem is largely determined by the effectiveness of the interaction of each ostis-system (including each ostis-community), each human with their external environment, as well as the quality and compliance of the external environment itself. Therefore, the main purpose of the OSTIS Ecosystem is to improve the quality of the information environment for all entities that are part of the OSTIS Ecosystem. In other words, the OSTIS Ecosystem provides support for the Information Ecology of human society.

ostis-community
⇒ *subdividing**:
{• *minimal ostis-community*
• *ostis-systems collective*
}

Each human included in the OSTIS Ecosystem has an one-to-one correspondence with their personal assistant in the form of a personal ostis-assistant. Thus, the number of personal ostis-assistants included in the OSTIS Ecosystem coincides with the number of humans included in the OSTIS Ecosystem. An example of humans and their corresponding personal ostis-assistants is shown in Figure 1.

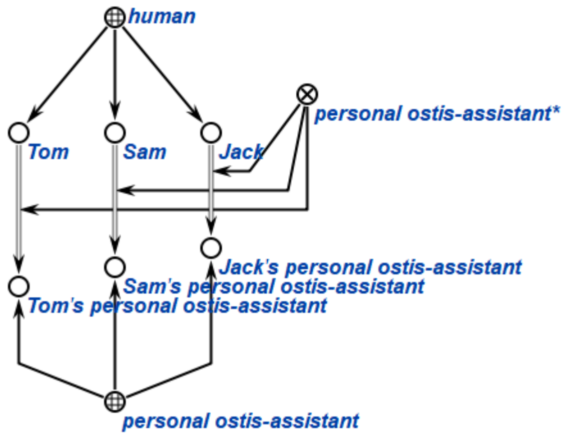

human
personal ostis-assistant*
Tom
Sam
Jack
Jack's personal ostis-assistant
Sam's personal ostis-assistant
Tom's personal ostis-assistant
personal ostis-assistant

Figure 1. Jack, Tom, and Sam as humans and their corresponding personal ostis-assistants

A collective consisting of a human and a corresponding personal ostis-assistant is actually a minimal ostis-community. An example of minimal ostis-communities is shown in Figure 2.

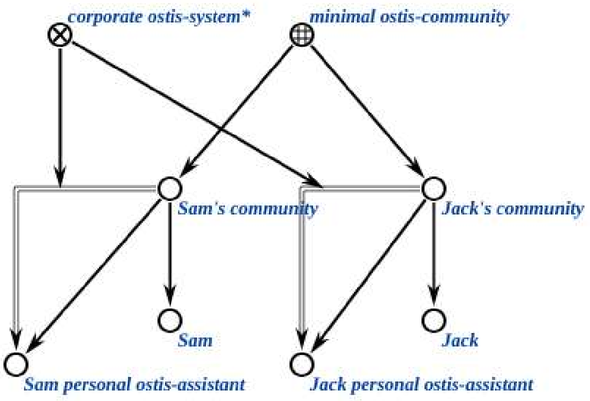


Figure 2. Samś and Jackś communities as objects of the *minimal ostis-community* class

Since, formally, non-minimal ostis-communities include not humans but personal ostis-assistants corresponding to them, all ostis-communities, except minimal ostis-communities, are collectives of ostis-systems.

The corporate ostis-system is the central ostis-system that coordinates, organizes, and supports the evolution of the activities of the members from the corresponding ostis-community. The corporate ostis-system is a representative of the corresponding ostis-community in other ostis-communities of which it is a member. An example of the ostis-system of the corporate community is shown in Figure 3.

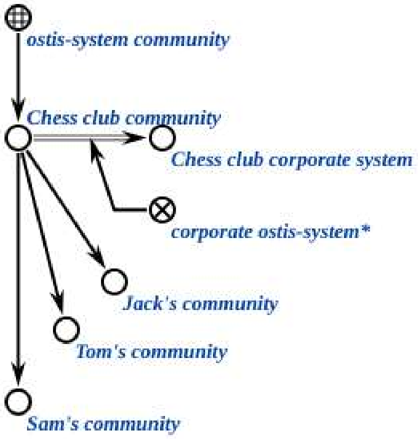


Figure 3. The chess club community with a corporate ostis-system

The main purpose of the OSTIS Ecosystem Corporate System is to organize common interaction in the performance of various types and areas of human activity, which can be either fully automated, or partially automated, or not automated at all. It follows from this that the knowledge base of the OSTIS Ecosystem Corporate System should contain the General formal theory of human activity, which includes a typology of types and areas of human activity, as well as a general methodology for this activity.

Activities in the field of Artificial Intelligence carried out on the basis of the OSTIS Technology
⇒ *core product**:
OSTIS Ecosystem
⇒ *subproject**:
- *OSTIS Metasystem Project*
- *Abstract sc-machine software implementation project*
- *Universal sc-computer development project*

The product of human activity in the field of Artificial Intelligence, carried out on the basis of OSTIS Technology, is not just a set of ostis-systems for various purposes but an Ecosystem consisting of interacting ostis-systems and their users. The typology of ostis-systems that are agents of the OSTIS Ecosystem is represented below.

ostis-system, which is an agent of the OSTIS Ecosystem
- ⊃ *personal ostis-assistant*
- ⊃ *corporate ostis-system*
- ⊃ *ostis-portal of scientific and technical knowledge*
- ⊃ *ostis-system of design automation*
- ⊃ *ostis-system of production automation*
- ⊃ *ostis-system of educational activities automation*
  - ⊃ *learning ostis-system*
  - ⊃ *corporate ostis-system of the virtual department*
- ⊃ *ostis-system of business automation*
- ⊃ *ostis-system of control automation*
  - ⊃ *ostis-system of project management of the appropriate type*
  - ⊃ *ostis-system of sensomotor coordination when performing a certain type of complex actions in the external environment*
    - ⊃ *ostis-system of self-driving control*

C. Purpose of creating the OSTIS Ecosystem

The OSTIS Ecosystem is a self-developing network of ostis-systems that provides comprehensive automation of various types and areas of human activity. A special place among the ostis-systems that are part of the OSTIS Ecosystem is occupied by corporate ostis-systems, through which the coordination and evolution of the activities of some groups of ostis-systems and their users is carried out. The main purpose of corporate ostis-systems is to localize the knowledge bases of the indicated groups of ostis-systems, transfer them from virtual to real status, and automate their evolution.

The OSTIS Ecosystem is the next stage in the development of human society, providing a significant increase in the level of public (collective) intelligence by transforming human society into an ecosystem consisting of humans and semantically compatible intelligent systems. The OSTIS Ecosystem is a proposed approach to the implementation of a smart society, or Society 5.0, built on the basis of the OSTIS Technology.

The super-purpose of the OSTIS Ecosystem is not just a comprehensive automation of all types of human activity (of course, only those activities whose automation is appropriate) but also a significant increase in the level of intelligence of various human (more precisely, human-machine) communities and the entire human society as a whole.

V. Integration of heterogeneous information resources and services in the OSTIS Ecosystem

It is very important to design not only the OSTIS Ecosystem itself as a form of implementation of Society 5.0 but also the process of a phased transition from a modern global network of computer systems to a global network of ostis-systems (i.e. to the OSTIS Ecosystem).

Within such a transitional period, ostis-systems can play the role of system integrators for various resources and services implemented by modern computer systems, since the level of intelligence of ostis-systems allows them to specify the integrated computer systems with any level of detail and, therefore, quite adequately "understand" what each of them knows and/or can. Also, ostis-systems are able to coordinate the activities of a third-party resource and service with sufficient quality and provide a "relevant" search for the desired resource and service. In addition, the systems can play the role of intelligent help systems – assistants and consultants on the effective operation of various computer systems with complex functionality, having a user interface with non-trivial semantics and used in complex subject domains. Such intelligent help systems can be made intelligent intermediaries between the respective computer systems of their users.

At the first stages of the transition to Society 5.0, there is no need to convert all modern automation systems into ostis-systems for certain types and areas of human activity. However, ostis-systems should take on a coordinating and connecting role due to the high level of their interoperability. The ostis-systems must learn to either fulfill the mission of an active interoperable superstructure over various modern automation tools or set problems that are feasible for modern automation tools, ensuring their direct participation in solving complex problems and organizing management of the interaction of various automation tools in the process of collectively solving complex complex problems.

The most important feature in the development of ostis-systems is that the development of an ostis-system is actually reduced to the development of its knowledge base. When developing the components of the problem solver and the interface, their features are taken into account, however, the general mechanism for making any changes to the ostis-system becomes a single one [23].

VI. Semantically compatible intelligent ostis-portals of scientific knowledge

Without the General formal theory of intelligent systems, it is impossible to build a set of methods and tools that provide comprehensive support for the development of intelligent computer systems for various purposes and with a different set of skills that intelligent computer systems may have but not necessarily each of them. At the same time, it is important not only to build a General theory of intelligent systems and bring it to a strict formal level but also to bring the representation of such a formal theory to the level of the knowledge base of the corresponding scientific knowledge portal.

The purposes of the intelligent portal of scientific knowledge are:

- accelerating the immersion of each human in new scientific areas while constantly maintaining a common holistic picture of the World (educational purpose);
- fixing new scientific results in a systematic way so that all the main connections between new results and known ones are clearly indicated;
- automating coordination of work on reviewing new results;
- automating the analysis of the current state of the knowledge base.

The creation of intelligent portals of scientific knowledge, providing an increase in the pace of integration and negotiation of different points of view, is a way to significantly increase the pace of evolution of scientific and technological activities. Compatible portals of scientific knowledge, implemented in the form of ostis-systems included in the OSTIS Ecosystem, are the basis of new principles for organizing scientific activity, in which the results of this activity are not articles, monographs, reports, and other scientific and technical documents but fragments of the global knowledge base, the developers of which are freely formed scientific collectives, consisting of specialists in the relevant scientific disciplines. With the help of scientific knowledge portals, both the process of reviewing new scientific and technical information coming from scientists to the knowledge bases of these portals is coordinated, and the process of coordinating different points of view of scientists (in particular, the introduction and semantic correction of concepts, as well as the introduction and correction of terms corresponding to different entities).

The implementation of a family of semantically compatible scientific knowledge portals in the form of compatible ostis-systems that are part of the OSTIS Ecosystem involves the development of a hierarchical system of semantically consistent formal ontologies corresponding to various scientific and technical disciplines, with a clearly defined inheritance of the properties of the described entities and with clearly defined interdisciplinary connections, which are described by connections between the corresponding formal ontologies and the subject domains specified by them.

An example of a scientific knowledge portal built in the form of an ostis-system is the OSTIS Metasystem, which contains all currently known knowledge and skills that are part of the OSTIS Technology.

VII. Semantically compatible intelligent corporate ostis-systems for various purposes

The corporate ostis-system allows monitoring, analyzing, and gradually automating all data processing processes within the ostis-community. Such a system operates according to the following principles:

- intelligent subsystems (agents) organize the data structure in such a way that up-to-date information is always available and outdated information is automatically archived or deleted in accordance with data storage and protection laws;
- requests to the system are executed in the form of simple instructions, the system helps managers enter the necessary information, performs partial or complete automation of updating information from databases available via the Internet;

- intelligent subsystems perform structuring and classification of documents and information to make quick and correct decisions, automatically process documents and available databases to select key information needed now and in the future;
- the existing system environment in the enterprise can be easily connected to the system through open interfaces; all information remains available;
- all key data systems are synchronized with the main system; data is constantly compared with each other to avoid loss;
- all information is available in the knowledge base, which is the source of data for workflows, reporting, and comprehensive checks.

Thus, the proposed platform allows representing all information about the ostis-community in a single, holistic way. The advantages of introducing the proposed system are:

- help in collecting and evaluating information without intentional misrepresentations or human error;
- providing full control over own data;
- the system provides only high-quality, reliable, and up-to-date data;
- digital representation of all community processes provides integrated information processing within the community, which gives full transparency of management, facilitates access to all information and its analysis;
- thanks to the support of intelligent subsystems, all the necessary data from documents, processes, and external sources can be extracted, structured, and properly evaluated.

Corporate ostis-systems can be applied in various areas: medicine and healthcare, educational activities of a wide profile, insurance business, industrial activities, administrative activities, real estate, transport, etc.

VIII. Personal user assistants

Society must turn its "face" to each human, be responsible and really contribute to each human personally, adapting to their characteristics, to ensure:

- the maximum level of physical health, activity, and longevity;
- the maximum level of physical comfort, personal space, material consumption;
- the maximum level of social, administrative, and legal comfort.

For this, the following must be carried out:

- personal monitoring of each human in all directions;
- diagnostics and elimination of unwanted deviations;
- provision of timely personal assistance in clarifying the directions of further evolution of each personality.

It is necessary to move from the provision of services in solving various problems at the initiative of the humans who have encountered these problems to the timely detection of the possibility of these problems and to appropriate prevention. This is possible only if there is a clear system organization of personal monitoring.

The client is not required to know the set of services from which they must choose the functionality that suits them. For the client, a set of semantically compatible services should be located "behind the scenes". Therefore, all information resources and services used by the client must be semantically compatible. The choice of a resource or service suitable for the user should be made by their personal assistant.

The personal assistant must take into account that the roles of clients can change, expand, as well as their interests and purposes. At the same time, all personal assistants must be semantically compatible in order to understand each other and also have the ability to independently interact within various corporate systems, representing the interests of their clients.

Personal ostis-assistant is an ostis-system, which is a personal assistant of the user within the OSTIS Ecosystem. This system provides the opportunity to:

- analysis of user activity and the formation of recommendations for its optimization;
- adaptation to the user's mood, their personal qualities, the general environment, the problem that the user most often solves;
- permanent training of the assistant itself in the process of solving new problems, while learning is potentially unlimited;
- conduct a dialog with the user in natural language, including in speech form;
- answer questions of various classes, and if something is not clear to the system, then it can ask counter questions;
- autonomously receive information from the entire environment, and not just from the user (in text or speech form).

At the same time, the system can both analyze available information sources (for example, on the Internet) and analyze the physical world around it, for example, surrounding objects or the user's appearance.

Advantages of a personal ostis-assistant:

- the user does not need to store different information in different forms in different places – all information is stored in a single knowledge base compactly and without duplication;
- due to unlimited learning, assistants can potentially automate almost any activity, not just the most routine;
- thanks to the knowledge base, its structuring, and information search tools in the knowledge base, the user can get more accurate information more quickly.

Personal assistants have a variety of purposes and can be used for a variety of categories of users (patient, legal service, administrative service, customer, consumer of various services).

IX. Knowledge Market

The evolution of ostis-systems and the OSTIS Ecosystem as a whole is a very complex creative, collective process, which in principle can only be partially automated. At the same time, high qualifications are required from humans participating in this process, the highest system culture at the level of deep knowledge of general systems theory, high mathematical culture – a culture of formalization, a high culture of convergence (discovering similarities, bringing them to formal analogies), a high culture of deep integration, high level of negotiation.

The evolution of ostis-systems and OSTIS Ecosystems as a whole comes down to the collective reengineering of knowledge bases of ostis-systems, which in turn comes down to:

- "manual" generation of proposed additional knowledge into the knowledge base of the specified ostis-system;
- "manual" generation of proposed changes to the current state of the knowledge base of the specified ostis-system;
- "manual" review of each submitted proposal;
- automatic appointment of competent and interested reviewers;
- automatic appointment of a sufficiently wide range of competent and interested specialists to approve the proposal received;
- automatic decision-making in relation to the received proposal based on the opinion of all involved experts and specialists.

In the knowledge base of each ostis-system, it is possible to record the entire process of discussing each proposal received, indicating the time points of all involved events, as well as the participants in each event (authors of proposals, authors of reviews, of polling participants). Each ostis-system, analyzing the process of using the knowledge stored by it during operation, can estimate the frequency of direct and indirect usage of this knowledge, i.e. can assess the degree of demand for this knowledge.

Therefore, in the future, the OSTIS Ecosystem with a sufficiently high degree of objectivity can assess the volume and significance of the contribution of each specialist to the development of the distributed knowledge base of the OSTIS Ecosystem. This is a fundamental basis for the formation of a fairly objective, honest knowledge market.

The knowledge proposed for review, approval, and publication, must be specified in the knowledge base of the corresponding ostis-system: ostis-system, atomic section of the knowledge base, date and time, author, type of publication. Copyright protection should not occur at the document level but at the content level of a knowledge base fragment.

Absolutely ideal solutions, including design ones, do not exist. Minimizing the degree of falsity can be ensured by:

- error-correction speed (other participants are less likely to use the erroneous fragment for their own purposes);
- improving the quality of analysis when making a decision (collective expertise, a larger number of experts involved, taking into account the level of expertise of an expert within a specific subject domain).

The implementation of the knowledge market will make it possible to make the transition from the classical representations of dictionaries and encyclopedias to a semantic network of specifications for all described entities. Such specifications will automatically determine the presence or absence of a synonymous sign within the technical state of the knowledge base for any new sign entered into the knowledge base.

X. Conclusion

The key direction in increasing the level of intelligence of individual intelligent cybernetic systems is the transition from individual intelligent cybernetic systems, absolutely independent of each other, to their universal communities, i.e. to multi-agent systems, independent agents of which are the indicated individual intelligent cybernetic systems. Within such systems, the possibility of communication of each agent with each one is provided, as well as the possibility of forming specialized collectives for the collective solution of complex collective problems. The implementation of the specified universal community of interoperable intelligent cybernetic systems is carried out in the form of the OSTIS Global Ecosystem.

The OSTIS Ecosystem is the basis for transferring the level of informatization of various areas of human activity to a fundamentally new level, as well as for integrating relevant projects: "Society 5.0", "Industry 4.0", "Smart House", "Smart City", "Knowledge market", and others. Without intelligent computer systems, all these projects are impossible.

The process of transition of the cybernetic systems community to next-generation intelligent cybernetic systems can take place gradually, where ostis-systems can play the role of coordinator of activities performed by other systems. Specified types of ostis-systems, such as semantically compatible intelligent ostis-portals of scientific knowledge, semantically compatible intelligent corporate ostis-systems, and personal user assistants, contribute to increasing the level of global community intelligence by increasing the level of interoperability.

The projects must be brought into a single coherent hierarchical system of interrelated projects covering the entire scope and diversity of human activity.

Acknowledgment

The author would like to thank the research group of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for its help in the work and valuable comments.

References

[1] F. W. Neiva, J. M. N. David, R. Braga, and F. Campos, “Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature,” *Information and Software Technology*, vol. 72, pp. 137–150, 2016.

[2] P. Lopes de Lopes de Souza, W. Lopes de Lopes de Souza, and R. R. Ciferri, “Semantic interoperability in the internet of things: A systematic literature review,” in *ITNG 2022 19th International Conference on Information Technology-New Generations*, S. Latifi, Ed. Cham: Springer International Publishing, 2022, pp. 333–340.

[3] Hamilton, Gunther, Drummond, and Widergren, “Interoperability - a key element for the grid and der of the future,” in *2005/2006 IEEE/PES Transmission and Distribution Conference and Exhibition*, 2006, pp. 927–931.

[4] P. Marrow, “Nature-inspired computing technology and applications,” *Bt Technology Journal - BT TECHNOL J*, vol. 18, pp. 13–23, 10 2000.

[5] G. Briscoe and P. De Wilde, “Self-organisation of evolving agent populations in digital ecosystems,” 01 2012.

[6] K. Kelly, *Out of Control: The New Biology of Machines, Social Systems, and the Economic World*. USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[7] H. Boley and E. Chang, “Digital ecosystems: Principles and semantics,” in *2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference*, 2007, pp. 398–403.

[8] M. Tsujimoto, Y. Kajikawa, J. Tomita, and Y. Matsumoto, “A review of the ecosystem concept — towards coherent ecosystem design,” *Technological Forecasting and Social Change*, vol. 136, pp. 49–58, 2018.

[9] H. Dong and F. K. Hussain, “Digital ecosystem ontology,” in *2007 IEEE International Symposium on Industrial Electronics*. IEEE, 2007, pp. 2944–2947.

[10] D. Cobben, W. Ooms, N. Roijakkers, and A. Radziwon, “Ecosystem types: A systematic review on boundaries and goals,” *Journal of Business Research*, vol. 142, pp. 138–164, 2022.

[11] M. Borgh, M. Cloodt, and G. Romme, “Value creation by knowledge-based ecosystems: Evidence from a field study,” *R&amp D Management*, vol. 42, pp. 150–169, 02 2012.

[12] R. Kapoor, “Ecosystems: broadening the locus of value creation,” *Journal of Organization Design*, vol. 7, no. 1, pp. 1–16, 2018.

[13] M. Hadzic, T. Dillon, and E. Chang, “Use of digital ecosystem and ontology technology for standardization of medical records,” in *2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference*, 2007, pp. 595–601.

[14] A. Elizarov, A. Kirillovich, E. Lipachev, and O. Nevzorova, “Digital ecosystem ontomath: Mathematical knowledge analytics and management,” in *Data Analytics and Management in Data Intensive Domains*, L. Kalinichenko, S. O. Kuznetsov, and Y. Manolopoulos, Eds. Cham: Springer International Publishing, 2017, pp. 33–46.

[15] S. D. Nagowah, H. Ben Sta, and B. Gobin-Rahimbux, “A systematic literature review on semantic models for iot-enabled smart campus,” *Applied Ontology*, vol. 16, no. 1, pp. 27–53, 2021.

[16] C. Reinisch, M. J. Kofler, and W. Kastner, “Thinkhome: A smart home as digital ecosystem,” in *4th IEEE International Conference on Digital Ecosystems and Technologies*. IEEE, 2010, pp. 256–261.

[17] E. G. Caldarola, A. Picariello, and A. M. Rinaldi, “An approach to ontology integration for ontology reuse in knowledge based digital ecosystems,” ser. MEDES ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1–8. [Online]. Available: https://doi.org/10.1145/2857218.2857219

[18] G. Briscoe and P. De Wilde, “Digital ecosystems: Evolving service-oriented architectures,” 12 2007.

[19] G. Briscoe, S. Sadedin, and P. De Wilde, “Digital ecosystems: Ecosystem-oriented architectures,” *Natural Computing*, vol. 10, no. 3, pp. 1143–1194, 2011.

[20] W. Li, Y. Badr, and F. Biennier, “Digital ecosystems: Challenges and prospects,” in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, ser. MEDES ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 117–122.

[21] Vladimir Golenkov and Natalia Guliakina and Daniil Shunkevich, *Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[22] (2022, Nov) Ostis Metasystem. [Online]. Available: https://ims.ostis.net

[23] V. Golenkov, N. Gulyakina, I. Davydenko, and D. Shunkevich, “Semanticheskie tekhnologii proektirovaniya intellektual’nyh sistem i semanticheskie associativnye komp’yutery [Semantic technologies of intelligent systems design and semantic associative computers],” *Otkrytye semanticheskie tehnologii proektirovanija intellektual’nyh sistem [Open semantic technologies for intelligent systems]*, pp. 42–50, 2019.

Принципы реализации экосистемы интеллектуальных компьютерных систем нового поколения

Загорский А. Г.

В работе рассмотрена архитектура экосистемы интеллектуальных компьютерных систем на основе Технологии OSTIS. Уточнена формальная трактовка таких понятий, как ostis-система, ostis-сообщество, выделена типология ostis-систем, что в совокупности позволяет определить структуру Экосистемы OSTIS. Полученные результаты могут быть применены при реализации таких проектов, как “Общество 5.0”, “Industry 4.0”, “Умный дом”, “Умный город”, “Рынок знаний”.

Received 30.10.2022

Metasystem of the OSTIS Technology and the Standard of the OSTIS Technology

Kseniya Bantsevich
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: ksusha.bantsevich@gmail.com

***Abstract*—In the article, an approach to automating the processes of creation, development, and application of standards based on the OSTIS Technology is proposed. The general problems related to the development and usage of modern standards in various fields are considered. Standardization of intelligent computer systems is proposed, as well as standardization of methods and means of their design within the proposed approach.**

***Keywords*—Standard, standard of intelligent computer systems, Metasystem of the OSTIS Technology, Standard of the OSTIS Technology.**

I. Introduction

Each developed sphere of human activity is based on a number of standards that formally describe its various aspects — a concepts system (including terminology), typology, sequence of actions performed in the process of applying appropriate methods and tools, and much more [1].

Standards in a wide variety of fields are the most important knowledge type, the main purpose of which is to ensure the compatibility of various activities. Despite the development of information technologies, currently, the vast majority of standards are represented either in the form of traditional linear documents or in the form of web resources containing a set of static pages connected by hyperlinks. In order for standards to fulfill their main function, they must be constantly improved. Due to the need for their permanent development, the current design of standards does not meet modern requirements.

II. Analysis of existing problems and approaches to their solutions

The current design of standards has a number of disadvantages that prevent the effective and competent usage of standards in various fields [2], [3]:

- duplication of information within the document describing the standard;
- the complexity of maintaining the standard itself due, among other things, to the duplication of information, in particular, the complexity of changing terminology;
- the problem of internationalization of the standard — in fact, the translation of the standard into several languages leads to the need to support and coordinate independent versions of the standard in different languages;
- the complexity of studying and applying standards;
- and others.

The listed problems are mainly related to the form of representation of standards. To solve these problems, standards should be designed in the form of intelligent reference systems that are able to answer a variety of questions. Thus, it is advisable to design standards in the form of knowledge bases, corresponding to intelligent reference systems. This approach makes it possible to significantly automate the processes of developing the standard and its application [4], [5].

Another urgent problem in the field of creating and applying standards for comprehensive technologies is the problem of their incompatibility, since different aspects of technology can be standardized by different standards that are incompatible with each other due to the inconsistency of the system of concepts and terms.

III. Proposed approach

Currently, *Informatics* is overcoming the most important stage of its development – the transition from data informatics (data science) to knowledge informatics (knowledge science), where attention is focused on semantic aspects of the representation and processing of *knowledge*.

Without a fundamental analysis of such a transition, it is impossible to solve many problems related to the management of *knowledge*, the economy of *knowledge*, *semantic compatibility* of *intelligent computer systems*.

To solve the above problems, it is proposed to use the *OSTIS Technology*, the main feature of which is the focus on the usage of next-generation computers specifically designed for the implementation of semantically compatible hybrid *intelligent computer systems* [6].

From a semantic point of view, each standard is a hierarchical ontology that clarifies the structure and systems of concepts of their corresponding subject domains, which describes the structure and functioning of either a certain class of technical or other artificial systems, or a certain class of organizations, or a certain type of activity. This approach provides obvious advantages in terms of automating the processes of harmonization and usage of standards [7], [8].

As part of this work, the experience of using this *technology* when designing the *Standard of the OSTIS Technology* will be considered. The suggested *Standard of the OSTIS Technology* is designed in the form of a *family of knowledge base sections* of a special intelligent computer *OSTIS Metasystem* (Intelligent MetaSystem for ostis-systems) [9], which is built based on the *OSTIS Technology* and represents a constantly improving intelligent *portal of scientific and technical knowledge*, which supports the permanent evolution of the *OSTIS Standard*, as well as the development of various *ostis-systems* (intelligent computer systems built based on the *OSTIS Technology*).

The *OSTIS Technology* is a complex of models, techniques, automated methods, and tools permanently developed within an open project, focused on ontological design, production, operation, and reengineering of semantically compatible hybrid *intelligent computer systems* capable of interacting independently with each other.

The represented *technology* is a technology of a fundamentally new level, which is conditioned by:

- the high quality of intelligent computer systems (*ostis-systems*) developed on its basis – their *semantic compatibility*, the ability to interact independently, the ability to adapt to users, and the ability to adapt (train) users themselves to interact with intelligent computer systems more effectively;
- the high quality of the *technology* itself – the ability to integrate the most diverse *knowledge types* and the most diverse *problem-solving models*, the inextricable connection between the processes of development of *intelligent computer systems* and professional training of developers.

The *OSTIS* Technology is based on the usage of unified semantic networks with a basic set-theoretic interpretation of their elements as a method of knowledge representation. This way of knowledge representation is called an *SC-code*, and the semantic networks, represented in the *SC-code*, are called *sc-graphs* (*sc-texts*, or *texts of the SC-code*). The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, can be *sc-arcs* or *sc-edges* depending on their orientation). The *Alphabet of the SC-code* consists of five basic elements, on the basis of which SC-code constructions of any complexity are built, including the introduction of more particular kinds of sc-elements (e.g., new concepts). The memory storing SC-code constructions is called semantic memory, or *sc-memory*.

The key feature of the *SC-code* is the joint usage of the mathematical apparatus of a graph theory and a set theory. This allows, on the one hand, ensuring the strictness and universatility of formalization tools and, on the other hand, ensuring the convenience of storing and processing information represented in this form.

Within the technology, several universal variants of visualization of the *SC-code* constructions are also proposed, such as *SCg-code* (graphic version), *SCn-code* (non-linear hypertextual version), *SCs-code* (linear string version).

Within this article, fragments of structured texts in the SCn-code [9] will often be used, which are simultaneously fragments of source texts of the knowledge base, which are understandable both to a human and to a machine. This allows making the text more structured and formalized while maintaining its readability. The symbol ":=" in such texts indicates alternative (synonymous) names of the described entity, which reveal in more detail some of its features.

The key features of the represented *technology* are:

- Complex nature, consisting in:
  - supporting the design of not only separate components of *intelligent computer systems* but also *intelligent computer systems* as a whole;
  - supporting not only the design but also the entire *life cycle* of intelligent computer systems.
- Ensuring *semantic compatibility* of both components of *intelligent computer systems* and intelligent computer systems as a whole throughout their *life cycle*.
- Implementation of the *OSTIS Technology* in the form of a *next-generation intelligent computer system*, which is also built based on the *OSTIS Technology*.

OSTIS Technology
⇒ *class of products**:
ostis-system
:= [next-generation intelligent computer systems built using the OSTIS Technology]
⊂ *next-generation intelligent computer system*
∋ *OSTIS Metasystem*

IV. OSTIS Metasystem

The OSTIS Metasystem [9] is an intelligent computer system that provides:

- comprehensive information support for all stages of the *life cycle* of next-generation intelligent computer systems;
- automation of the design for all components of *next-generation intelligent computer systems*;
- comprehensive automation of all stages in the life cycle of *next-generation intelligent computer systems*.

The implementation form of the represented *OSTIS Metasystem* is the *OSTIS Technology*.

The *OSTIS Metasystem* is:

- a system of information and instrumental support for all stages of the life cycle of next-generation intelligent computer systems (*ostis-systems*) for various purposes;
- OSTIS Technology Knowledge Portal, providing:

□ □ coordination of work on the development of the OSTIS Technology;
□ □ automation of quality analysis of the OSTIS Standard.

That is, the OSTIS Metasystem is a project management system for the creation and development of the *OSTIS Standard*.

V. OSTIS Standard

The *Standard of the OSTIS Technology* [10] is a documentation of the OSTIS Technology, which is represented as the main part of the knowledge base of a special intelligent computer system designed to comprehensively support the life cycle of semantically compatible next-generation intelligent computer systems (the OSTIS Metasystem).

OSTIS Standard
:= [documentation of the *OSTIS Technology*]
:= [documentation of the Open Technology for ontological design, production, and operation of semantically compatible hybrid *intelligent computer systems*]
:= [description of the *OSTIS Technology* (Open Semantic Technology for Intelligent Systems), represented in the form of a family of sections in the *knowledge base of a special ostis-system* (system built based on the *OSTIS Technology*) in the internal language of *ostis-systems* and possessing sufficient completeness for the usage of this *technology* by developers of *intelligent computer systems*]
:= [complete description of the current state of the *OSTIS Technology*, represented in the form of a family of sections in the *knowledge base* built based on the *OSTIS Technology*]
:= [family of sections in the *knowledge base* of the *OSTIS Metasystem*, which is designed to provide comprehensive support for the ontological design of semantically compatible *hybrid intelligent computer systems*]
∈ *family of knowledge base sections*
    := [family of sections of the internal representation of the *ostis-system knowledge base* – an intelligent computer system built using the *OSTIS Technology*]
:= [fairly complete formal documentation of the current version of the *OSTIS Technology*, represented either as the main part of the knowledge base of the *OSTIS Metasystem* or as an external formal representation of this *knowledge base*]
:= [main part of the *OSTIS Metasystem* knowledge base describing the current version of the *OSTIS Technology*]
:= [formal text, the object of which is the *OSTIS Technology*, i.e. a text that is a fairly complete description of the current state of the *OSTIS Technology*]
:= [documentation of the *OSTIS Technology*, fully reflecting the current state of the *OSTIS Technology* and represented by the corresponding *family of knowledge base sections* of the *ostis-system*, which is focused on supporting the design, production, operation, and evolution (reengineering) of *ostis-systems*, as well as supporting the evolution of the *OSTIS Technology* itself and named the *OSTIS Metasystem*]
:= [family of sections that includes all sections of the *OSTIS Standard*]
⇒ *main sc-identifier**:
[OSTIS Standard]
    ⇐ *reduction**:
    [Standard of the *OSTIS Technology*]
        ⇐ *reduction**:
        [Standard of an Open *technology* for integrated support of the *life cycle* of semantically compatible *next-generation intelligent computer systems*]

It should be emphasized that the *OSTIS Standard* is not a description of a certain state of the *OSTIS Technology* but a dynamic information model of the evolution of this *technology*.

VI. Table of contents of the OSTIS Standard

One of the components of the *OSTIS Standard* is the *Table of contents of the OSTIS Standard*.

The *Table of contents of the OSTIS Standard* is a hierarchical list of sections included in the *OSTIS Standard*, with an additional specification of some sections indicating their alternative names. It is essential to emphasize that the hierarchy of sections of the *OSTIS Standard* does not mean that *sections* of a lower level of the hierarchy are part of the corresponding sections of a higher level. The relation between *sections* at different levels of the hierarchy means that a *section* at a lower level of the hierarchy is a *child* section in relation to the corresponding *section* at a higher level, i.e. a section that inherits the properties of the specified *section* at a higher level.

In contrast, each *part of the OSTIS Standard*, as well as the *OSTIS Standard* itself, is a *family of sections* (a set of sections) that are part of it.

VII. General structure of the OSTIS Standard

Consider the structure of the top-level *OSTIS Standard*:

- Part 1 of the OSTIS Standard.
  Introduction to next-generation intelligent computer systems
  - := [Analysis of the current state of *technologies of Artificial Intelligence* and setting the objective for creating a *complex* of compatible *Artificial Intelligence technologies* that provides support for the entire *life cycle of next-generation intelligent computer systems* and named an *OSTIS Technology*]
- Actual documentation of the *OSTIS Technology*
  - Part 2 of the OSTIS Standard.
    Semantic representation and ontological systematization of knowledge of the next-generation intelligent computer systems
    - := [Standard for information representation in ostis-systems]
    - := [Models of representation of knowledge and knowledge bases in ostis-systems]
  - Part 3 of the OSTIS Standard.
    Multi-agent problem solvers of next-generation intelligent computer systems
    - := [Standard of processes and methods of information processing in ostis-systems]
    - := [Knowledge processing models in ostis-systems (logical, production, functional, neural network, procedural and non-procedural, clear and fuzzy)]
  - Part 4 of the OSTIS Standard.
    Ontological interfaces models of next-generation intelligent computer systems
    - := [Standard of information resources and models for solving interface problems in ostis-systems]
- Standard of methods and means of life cycle support for ostis-systems
  - := [Standard of business processes and techniques, automatically implemented processes and methods, information tools and tools used to support the life cycle of ostis-systems]
  - Part 5 of the OSTIS Standard.
    Methods and means of designing next-generation intelligent computer systems
    - := [Techniques, methods, and tools for designing knowledge bases, problem solvers, and interfaces of ostis-systems]
  - Part 6 of the OSTIS Standard.
    Implementation platforms for next-generation intelligent computer systems
    - := [Methods and means of implementing ostis-systems (based on software platforms and specially designed computers for this purpose)]
  - Part 7 of the OSTIS Standard.
    Methods and means of reengineering and operation of next-generation intelligent computer systems
    - := [Methods and means of ostis-systems operating by end users, as well as their maintenance (maintenance of operability) and reengineering (updates, upgrades)]
- Part 8 of the OSTIS Standard.
  Ecosystem of next-generation intelligent computer systems and their users
  - := [Description of products created using the OSTIS Technology, the main of which is the *OSTIS Ecosystem*, semantically compatible and actively interacting ostis-systems and their users]
  - := [Theory of the OSTIS Ecosystem and its evolution]
- OSTIS Bibliography
  - := [Specification of *bibliographic sources* semantically close to the *OSTIS Technology*, in the context of their comparative analysis with the *OSTIS Standard*]

VIII. Key signs of the OSTIS Standard

The system of key signs of the OSTIS Standard is ordered in exact accordance with the Table of contents of the OSTIS Standard and is a clarification of the specified Table of contents by listing and explaining the key entities described in the sections of the Standard, and, first of all, those entities that are specified in the identifiers (names) of the sections of the OSTIS Standard.

The *System of key signs of the OSTIS Standard* is a complete addition to the Table of contents of the OSTIS Standard, since:

- the hierarchy and sequence of key characters clearly correspond to the hierarchy and sequence of sections of the standard;
- the system of key signs of the OSTIS Standard, as well as its Table of contents, is perceived (read) as a complete understandable text.

IX. Purpose of the OSTIS Standard

Since the *OSTIS Standard* is an integral part of the *OSTIS Metasystem* (the main part of its knowledge base), the main purpose of the *OSTIS Standard* is to ensure the most effective implementation of what the *OSTIS Metasystem* is designed for.

In addition, the most important direction of the *OSTIS Metasystem* and, accordingly, the most important direction of the application of the *OSTIS Standard* is their usage as a comprehensive integrated computer textbook in the specialty "Artificial Intelligence". For this purpose, a connection is established between the sections of the *OSTIS Standard* and the programs of various academic disciplines of the specified specialty. It is important to emphasize at the same time: the *OSTIS Standard* contains a fairly complete comparative analysis with various alternative approaches, i.e. in no case is limited to considering only the *OSTIS Technology*.

The *OSTIS Standard* is considered as a result of convergence and integration of various directions of *Artificial Intelligence*, which allows students and undergraduates to form a holistic view of the subject of *Artificial Intelligence*, rather than a mosaic representation in the form of a variety of disciplines (directions), the connections between which are not considered in detail and even more formally.

The *OSTIS Standard* is permanently and rapidly evolving. During the training of students and undergraduates, there are very significant changes in the current version of the *OSTIS Standard*.

Students and undergraduates are actively involved in the process of evolution of the *OSTIS Standard*, which ensures:

- formation of the necessary level of their qualifications in conditions of rapid moral aging of what they have already been taught;
- formation of the necessary skills that allow them to quickly adapt to the new conditions of this activity and, in particular, to new versions of relevant technologies in the process of real professional activity.

X. Analogues of the OSTIS Standard

Analogs of the *OSTIS Standard* include:

- any serious attempt to systematize the results obtained in the field of Artificial Intelligence to the current moment:
  - a textbook that fully reflects the current state of *Artificial Intelligence*;
  - a reference book containing fairly complete information about the current state of *Artificial Intelligence*.
- any attempt to move from particular formal models of various components from the intelligent computer systems of the general (combined, integrated) formal model of intelligent computer systems as a whole to the general theory of intelligent computer systems;
- any unification of technical solutions, elimination of the variety of forms of technical solutions in the development of intelligent computer systems;
- the first attempts to develop standards for *intelligent computer systems*, as well as *Artificial Intelligence technologies*, which, most often, are limited to the building of systems of corresponding concepts.

XI. Features of the OSTIS Standard

The *OSTIS Standard* is not just a systematization of the current state of results in the field of *Artificial Intelligence* – it is a systematization represented in the form of a general complex formal model of *intelligent computer systems* and a complex formal model to support their life cycle. Moreover, the text of the *OSTIS Standard* is the *main part of the knowledge base* of a special *intelligent metasystem*, which is focused on:

- support for the development of *intelligent computer systems* for various purposes;
- support for the evolution of the *OSTIS Standard*;
- support for the *training of specialists in the field of Artificial Intelligence*.

The *OSTIS Standard* is a dynamic text that permanently reflects new scientific and technical results obtained in the field of *Artificial Intelligence* within a *General theory of intelligent computer systems* and *General comprehensive technology for the development of intelligent computer systems*. It is important that new scientific and technical results are recorded promptly, i.e. minimizing the time interval between the moment of obtaining new results and the moment of integrating the description of these results into the *OSTIS Standard*. In the future, the authors of new scientific and technical results in the field of *Artificial Intelligence* will be interested in personally publishing (integrating) their results into the *OSTIS Standard*, i.e. becoming co-authors of the *OSTIS Standard* to ensure the necessary efficiency of such publication and the absence of distortion of its results. Dynamism of the *OSTIS Standard* and efficiency of integration into its structure of new scientific and technical results in the field of *Artificial Intelligence* does the *OSTIS Standard* always relevant and never obsolete.

Within the *OSTIS Standard*, there is no opposition between scientific and technical information, obtained in the field of Artificial Intelligence, and educational and methodological information, used for the training and self-training of specialists in the field of Artificial Intelligence. Information about what to learn should be "intertwined", integrated with information about how to learn.

It is important to note that the *OSTIS Standard*, unlike other standards, is a structured formal text that can be directly used not only by developers of *intelligent computer systems* but also by intelligent computer systems that automate the design of developed *intelligent computer systems* and support subsequent stages of their life cycle. Thus, the development of the *OSTIS Standard* is an integral part in the development of a set of information and instrumental support tools for the entire life cycle of *intelligent computer systems*, and these support tools of *life cycle of intelligent computer systems* become equal partners in the process of creating, operating, and maintaining *intelligent computer systems* due to their awareness (understanding) in *intelligent computer systems* and their *life cycle*.

Contensive, the *OSTIS Standard* covers not only the description of models of intelligent computer systems being developed but also the description of techniques, automated methods, and tools for supporting (automating) all stages of the life cycle of intelligent computer systems being developed.

The OSTIS Standard development project is focused on high rates of evolution of the *OSTIS Standard* thanks to automating the management of this project using the *OSTIS Metasystem*, which is a full participant in this project.

Building and structuring of text of the *OSTIS Standard* are focused on the maximum possible reduction of the language and cognitive barrier for its novice users. For this purpose, (1) various kinds of natural language notes and comments are used that have appropriate semantic connections with the entities being explained, as well as (2) various kinds of didactic knowledge indicating various analogies, differences, examples, principles underlying the described entities, etc.

XII. User of the OSTIS Standard

Consider the target audience of the *OSTIS Standard*.

OSTIS Standard
⇒ *users class**:
user of the OSTIS Standard
:= [target audience of the OSTIS Standard]
⇒ *subdividing**:
{• *developer of the ostis-system*
⊃ *developer of the OSTIS Metasystem*
⊃ *developer of the ostis-system knowledge base*
⊃ *developer of the OSTIS Standard*
⊃ *developer of the ostis-system problem solver*
⊃ *developer of the OSTIS Metasystem problem solver*
⊃ *developer of the ostis-system interface*
⊃ *developer of the OSTIS Metasystem interface*
⊃ *developer of ostis-systems implementation platforms*
• *potential developer of the ostis-system*
• *specialist in the field of Artificial Intelligence who wants to integrate their results into the general theory of next-generation intelligent computer systems and the corresponding comprehensive technology*
• *student or master student of the "Artificial Intelligence" specialty or another related specialty who wants to gain practical experience in the development of applied next-generation intelligent computer systems or in the development of an appropriate comprehensive technology*
}

developer of the OSTIS Metasystem
⇒ *subdividing**:
{• *developer of the OSTIS Standard*
• *developer of the OSTIS Metasystem problem solver*
• *developer of the OSTIS Metasystem interface*
}

XIII. Writing Team of the OSTIS Standard

To ensure permanent evolution, there are a number of requirements focused on the authors of the *OSTIS Standard*.

Authors of the *OSTIS Standard* should:

- Track and study new publications on the topics covered in the *OSTIS Standard*. Close sources for this are:
  - magazine issues;
  - conference materials:
    - organized by the 3WC Consortium;
    - on the integration of various AI directions;
  - standards in the field of AI;
  - publications, considering:
    - formal ontologies;
    - top-level ontologies;
    - semantic networks;
    - knowledge graphs;
    - graph databases;
    - semantic representation of knowledge;
    - convergence of different AI directions.
- To record the results of the study of new publications on topics close to the OSTIS Standard in the OSTIS Bibliography, as well as in the main text of the OSTIS Standard in the form of appropriate references, citations, comparative analysis.
- Track the current state of the total text of the OSTIS Standard, form proposals aimed at the development of the OSTIS Standard and at increasing the pace

of this development; actively participate in the discussion of the problems of OSTIS Technology development.
- To connect personal work on the OSTIS Standard with other forms of activity – scientific, educational, applied – in the maximum possible way.
- Indicate the authorship of their proposals to supplement and/or correct the current text of the OSTIS Standard.
- Participate in reviewing and approving proposals submitted by other authors of the OSTIS Standard.

The large volume of work on the creation and development of the *OSTIS Standard* and, accordingly, the *OSTIS Technology*, the complex nature of these works, which require deep convergence and integration of various directions of *Artificial Intelligence*, place high demands on the *Writing Team of the OSTIS Standard* in terms of motivation, the quality of the creative atmosphere, the level of *interoperability* of all team members, i.e. the level of ability to quickly and efficiently coordinate personal points of view.

Since the Project for the creation and development of the *OSTIS Standard* is open, anyone who follows the Rules for organizing the interaction of members of the OSTIS Standard Writing Team, sharing the purposes and objectives of developing such a Standard can become a member of the OSTIS Standard Writing Team.

The following key points of the *Rules for organizing the interaction of members of the OSTIS Standard Writing Team of the OSTIS Standard* are highlighted:

- collectively form tactical and strategic directions for the development of the OSTIS Standard and, accordingly, OSTIS Technology;
- collectively distribute tasks for the implementation of the approved directions for the development of the OSTIS Standard, taking into account (1) the scientific interests, qualifications and capabilities of each member of the Writing Team, (2) the priority of tasks and a sufficiently complete coverage of all priority tasks.

XIV. Editorial Board of the OSTIS Standard

Within the *Writing Team of the OSTIS Standard*, the *Editorial Board of the OSTIS Standard* is also distinguished.

The *Editorial Board of the OSTIS Standard* is a part of the *Writing Team of the OSTIS Standard*, which is the center of collegial decision-making on the main directions of the development of the OSTIS Standard and, accordingly, the OSTIS Technology, in order to clarify the relevant priorities and terms. The *Editorial Board of the OSTIS Standard* is also responsible for the formation and implementation of strategic directions for the development of the OSTIS Standard and, in particular, for the selection and appointment of *responsible executors for sections of the OSTIS Standard*.

The main activities of the *Editorial Board of the OSTIS Standard* are:

- ensuring the integrity and improving the quality of the constantly developing OSTIS Technology, as well as a fairly accurate description (documentation) of each current version of this technology;
- ensuring clear control of compatibility of OSTIS Technology versions as a whole, as well as versions of various components of this technology;
- constantly clarifying the degree of importance for various directions of OSTIS Technology development for each current moment;
- formation and constant refinement of the plan for tactical and strategic development of the *OSTIS Technology* itself, as well as complete documentation of this Technology in the form of the *OSTIS Standard*. At the same time, we emphasize that this documentation is an integral part of the *OSTIS Technology*.

XV. Requirements for the OSTIS Standard

The *OSTIS Standard* must meet the following requirements:

- the concepts introduced (including didactic relations) should be clearly explained and/or defined in the relevant section of the OSTIS Standard;
- providing the possibility of step-by-step formalization of information, starting from nl-texts, which can later be written in a formal language;
- a clear logical and semantic specification of each subject domain considered in the OSTIS Standard. The named specification should reflect both the internal structure of the subject domain (the roles of its key elements) and the connections with other subject domains;
- convergence, ("seamless") integration of various *knowledge* types describing a variety of entities, which, in particular, include knowledge of all kinds;
- integrity, completeness, connectivity:
  - lack of information holes;
  - a fairly complete specification of all entities;
  - consistency of basic identifiers (terms), absence of synonyms and homonyms.
- absence of information excesses and information garbage;
- clear semantic stratification – each fragment of the knowledge base should have its own semantic "shelf" (no duplication);
- strict logical sequence of the text (all entities used must be introduced either in a given subject domain or in a higher-level subject domain);
- unification of stylistics – the text should not cause difficulties for its understanding;

- extended bibliography and comparative analysis;
- strict compliance with and improvement of the rules for identification and specification of the described entities;
- a sufficiently detailed specification of each introduced concept in the relevant subject domain.

XVI. Rules for the construction of the OSTIS Standard

As part of the development of the *OSTIS Standard*, the *General rules for the construction of the OSTIS Standard* and the *Particular rules for the construction of the OSTIS Standard* are distinguished.

General rules for the construction of the OSTIS Standard

:= [principles underlying the structuring and design of the OSTIS Standard]

Let us consider the main conditions:

- The main form of representation of the *OSTIS Standard* as a complete documentation for the current state of the *OSTIS Technology* is the *internal representation* of the main part from the *knowledge base* of the special intelligent computer *OSTIS Metasystem*, which ensures the usage and evolution (permanent improvement) of the *OSTIS Technology*. This representation of the *OSTIS Standard* provides effective semantic navigation through the contents of the *OSTIS Standard* and the ability to ask the *OSTIS Metasystem* a wide range of non-trivial questions about the most diverse details and subtleties of the *OSTIS Technology*.
- In addition to the representation of the *OSTIS Standard* in the internal *language of knowledge representation*, the external form of the representation of the *OSTIS Standard* in the *external language of knowledge representation* is also used. At the same time, the specified external representation of the *OSTIS Standard* should be structured and designed so that the reader can easily "manually" find almost any *information* of interest in this text. The *SCn-code* is used as the formal language of the external representation of the *OSTIS Standard*.
- The *OSTIS Standard* has an ontological structuring, i.e. it is a hierarchical system of related *formal subject domains* and their corresponding *formal ontologies*. This ensures a high level of stratification of the *OSTIS Standard*.
- Each concept used in the *OSTIS Standard* has its own place within this Standard, its own *subject domain* and its corresponding *ontology*, where this concept is considered (investigated) in detail, where all the basic information about this *concept*, about its various properties, is concentrated.
- The *OSTIS Standard* also includes files of information constructions that are not *SC-code* constructions (including sc-texts belonging to various natural languages). Such files allow formally describing the syntax and semantics of various external languages in the knowledge base, as well as also allow including in the knowledge base various kinds of explanations and notes addressed directly to users and helping them to understand the formal text of the knowledge base.
- From a semantic point of view, the *OSTIS Standard* is a hierarchical system of formal models of *subject domains* and their corresponding *formal ontologies*.
- From a semantic point of view, the *OSTIS Standard* is a large *refined semantic network*, which, accordingly, has a non-linear character and which includes signs of any types of entities described (material entities, abstract entities, concepts, connections, structures), as well as, accordingly, contains connections between all these types of entities (in particular, connections between connections, connections between structures).
- The *OSTIS Standard* is a hierarchical system of *subject domains* and their corresponding *ontologies* specifying these *subject domains*. Each of the *subject domains* describes the corresponding classes of research objects with the maximum possible degree of detail determined by a set of *relations* and *parameters* indicated on the classes of *research objects*. On a set of *subject domains*, the *private subject domain** relation is set, which indicates the direction of inheritance of properties for research objects considered in different *subject domains*.
- Each *section of the OSTIS Standard* may contain the *knowledge* that is part of the *subject domain and ontology*, which is either fully represented by the specified *section* or partially represented in the form of a specification of one or more specific research objects.
- Synonymy and homonymy of the main sc-identifiers within each family is not allowed.
- The specification of each subject domain and each section should have a sufficient degree of completeness. At a minimum, the role of each concept used in it should be specified for each subject domain.
- The *OSTIS Standard* itself is an internal *semantic representation* of the main part from the knowledge base of the *OSTIS Metasystem* in the internal semantic language of *ostis-systems* (this language is called an *SC-code* – Semantic Computer Code).

In addition to the *General Rules for the construction of the OSTIS Standard*, in the *OSTIS Standard*, descriptions of various particular (specialized) rules for constructing (formatting) various types of fragments of the *OSTIS Standard* are provided. These types of fragments include the following ones:

- *sc-identifier*
  - := [external identifier of the internal sign (*sc-element*) included in the *ostis-system knowledge base*]
  - := [*information construction* (most often, a string of characters) that provides unambiguous identification of the corresponding entity described in the *ostis-systems of knowledge bases* and is, most often, a name (term) corresponding to the entity being described, a name denoting this entity in the external texts of *ostis-systems*]
- *sc-specification*
  - := [semantic neighborhood]
  - := [semantic neighborhood of the corresponding *sc-element* (an internal sign stored in the memory of the *ostis-system* as part of its *knowledge base*, represented in the internal language of the *ostis-systems*)]
  - := [semantic neighborhood of some *sc-element* stored in *sc-memory* within the current state of this *sc-memory*]
- *sc-construction* $\setminus$ *sc-specification*
  - := [*sc-construction* (construction of the *SC-code* – the internal language of *ostis-systems*), which is not an *sc-specification*]
- *ostis-system file* $\setminus$ *sc-identifier*
  - := [*ostis-system file*, which is not an sc-identifier]

It is also important to note that among the particular rules for building *sc-constructions* there are *Rules for building ostis-systems knowledge bases*. These rules are aimed at ensuring the integrity of the ostis-systems knowledge bases, (1) the <u>relevance</u> (necessity) of the knowledge included in each knowledge base, and (2) integrity of the knowledge base itself, i.e. the sufficiency of the knowledge included in each knowledge base for the effective functioning of the corresponding ostis-system.

XVII. Directions of development of the OSTIS Standard

OSTIS Standard

⇒ *general directions of development**:

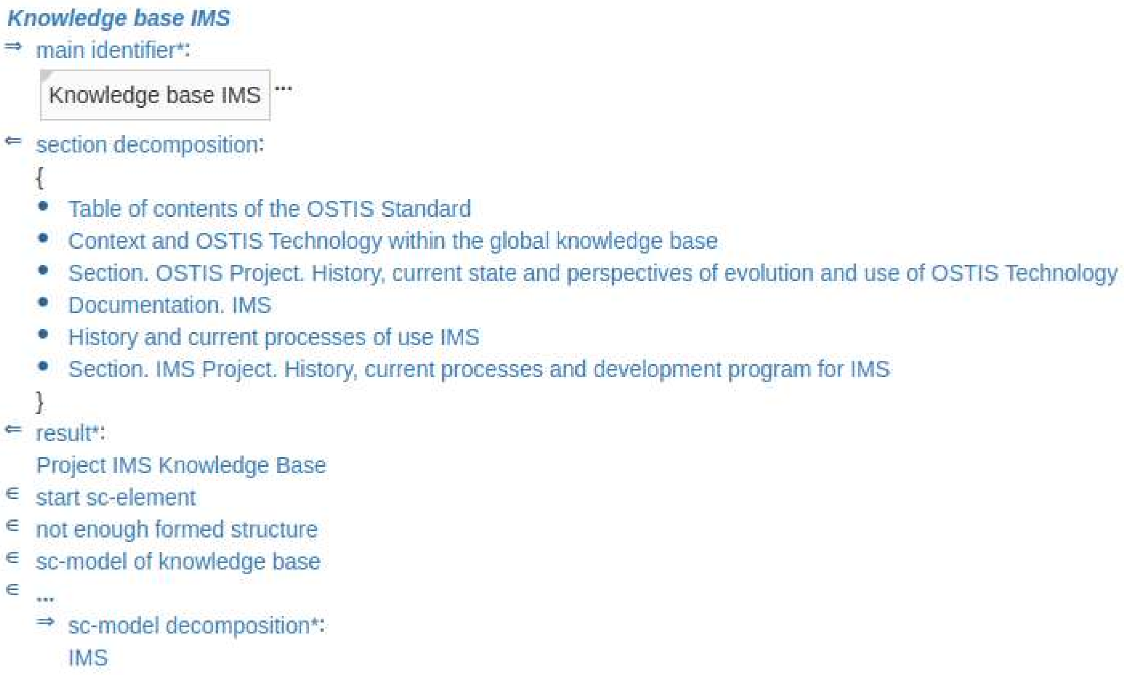
{
- [To include in the OSTIS Standard sufficiently detailed rules for the construction (design) of sc-identifiers and sc-specifications of various types of entities, as well as various types of files of ostis-systems]
- [At each stage, clearly distribute the work on the development of various sections of the OSTIS Standard]
- [All the tools that are part of the OSTIS Technology should be described (specified) in sufficient detail in the form of appropriate ontological models that have a clear semantic connection with the corresponding ontologies and related subject domains that are part of the OSTIS Standard]
- [To ensure sufficient <u>completeness</u> of the sc-specification of <u>all</u> entities under consideration]
- [Significantly expand the OSTIS Bibliography]
- [Constantly monitor the synonymy/homonymy of sc-identifiers]
- [Constantly improve quality control of work on the development of the OSTIS Standard]
- [It is necessary to constantly analyze the publications of other authors on issues close to the subject of the OSTIS Standard and to record in the OSTIS Standard the results of a comparative analysis of the point of view, represented in the OSTIS Standard, with the points of view of other authors by including specifications of relevant bibliographic sources with useful quotes in the OSTIS Standard]

}

XVIII. Advantages of the OSTIS Standard

The *OSTIS Standard* is an example of the transition to a fundamentally new form of representation and publication of *scientific and technical information*, research results – not just to the form of an electronic document but to the form of a semantically structured electronic document that is part of the *knowledge base* for the relevant *scientific and technical discipline*. This significantly increases the efficiency of using *scientific and technical information* accumulated by a human, since the user of this information can not only view (read) it but also interact with the intelligent computer system, which becomes a partner in using the information they need.

The *Project of creation and development of the OSTIS Standard* is a prototype of a fundamentally new approach to the organization of *scientific and technical activity* within each *scientific discipline*. This activity is implemented in the form of an open project aimed at developing the *knowledge base* of the intelligent knowledge portal in the relevant scientific and technical discipline. Such a level of formalization of *scientific and technical information*, which is understandable not only to specialists but also to *intelligent computer systems*, significantly increases the efficiency and expands the application areas for this information in *intelligent computer systems*. For example,



Knowledge base IMS
⇒ main identifier*:
Knowledge base IMS ...
⇐ section decomposition:
{
- Table of contents of the OSTIS Standard
- Context and OSTIS Technology within the global knowledge base
- Section. OSTIS Project. History, current state and perspectives of evolution and use of OSTIS Technology
- Documentation. IMS
- History and current processes of use IMS
- Section. IMS Project. History, current processes and development program for IMS

}
⇐ result*:
Project IMS Knowledge Base
∈ start sc-element
∈ not enough formed structure
∈ sc-model of knowledge base
∈ ...
⇒ sc-model decomposition*:
IMS

Figure 1. A home page of the *OSTIS Metasystem* knowledge base

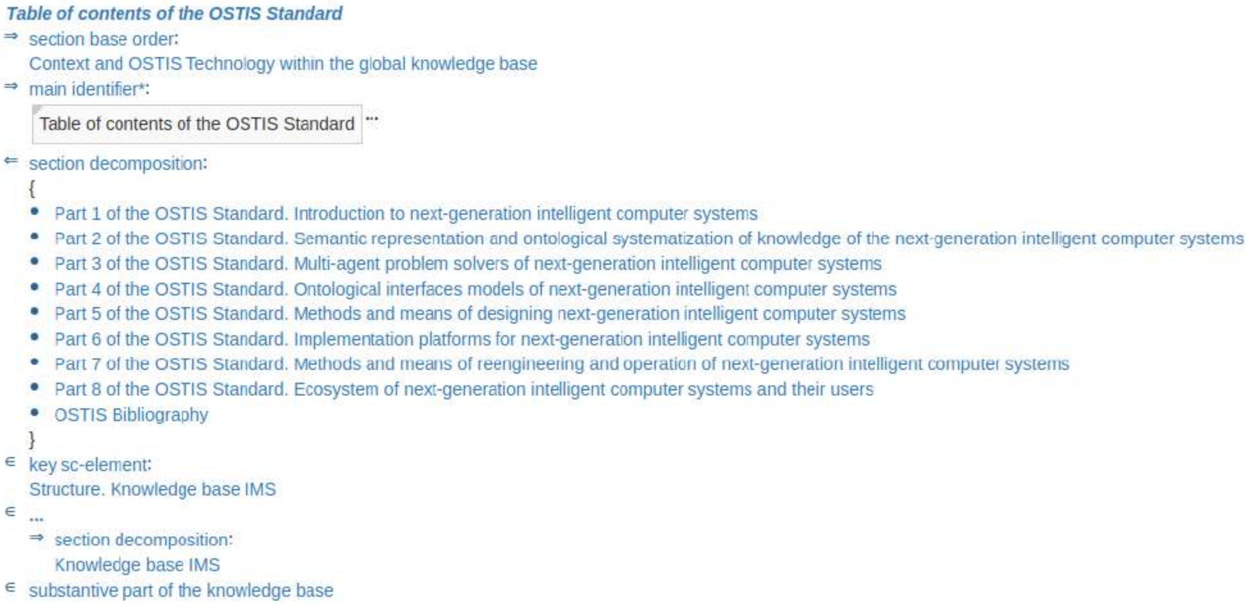


Table of contents of the OSTIS Standard
⇒ section base order:
Context and OSTIS Technology within the global knowledge base
⇒ main identifier*:
Table of contents of the OSTIS Standard ...
⇐ section decomposition:
{
- Part 1 of the OSTIS Standard. Introduction to next-generation intelligent computer systems
- Part 2 of the OSTIS Standard. Semantic representation and ontological systematization of knowledge of the next-generation intelligent computer systems
- Part 3 of the OSTIS Standard. Multi-agent problem solvers of next-generation intelligent computer systems
- Part 4 of the OSTIS Standard. Ontological interfaces models of next-generation intelligent computer systems
- Part 5 of the OSTIS Standard. Methods and means of designing next-generation intelligent computer systems
- Part 6 of the OSTIS Standard. Implementation platforms for next-generation intelligent computer systems
- Part 7 of the OSTIS Standard. Methods and means of reengineering and operation of next-generation intelligent computer systems
- Part 8 of the OSTIS Standard. Ecosystem of next-generation intelligent computer systems and their users
- OSTIS Bibliography

}
∈ key sc-element:
Structure. Knowledge base IMS
∈ ...
⇒ section decomposition:
Knowledge base IMS
∈ substantive part of the knowledge base

Figure 2. A Table of contents of the OSTIS Standard

an intelligent knowledge portal on a technical discipline naturally becomes an *intelligent system for automating the design* of technical systems of the appropriate class.

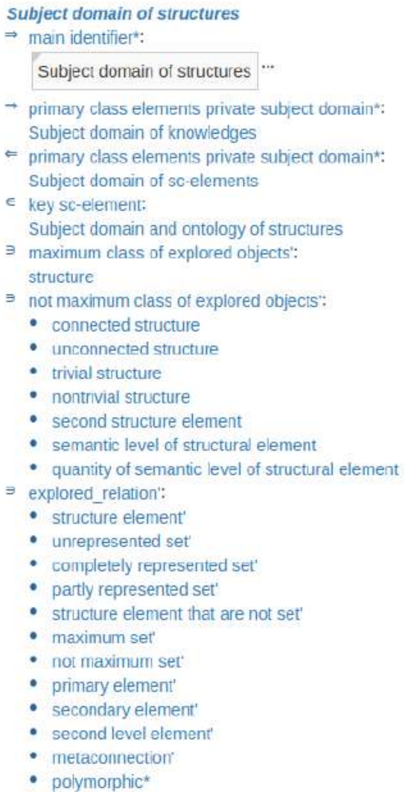
Another important advantage is the fact that the *OSTIS Standard* is a prototype of next-generation textbooks that have a clear logical-semantic structuring and stratification of educational material, as well as a set-theoretic and logical classification of all concepts used. Therefore, the usage of the *OSTIS Standard* not only as the basis of an intelligent automation system for integrated support of the *next-generation intelligent computer systems life cycle* but also as a comprehensive textbook for the training of young specialists in the field of *Artificial Intelligence* is a prototype of the widespread usage of various intelligent portals of scientific and technical knowledge as comprehensive textbooks for the training of young specialists in relevant specialties. This will significantly improve the quality of education, which should not lag behind the development of relevant scientific and technical areas but should become an integral part of this development.

XIX. OSTIS Standard as the main part of the knowledge base of the OSTIS Metasystem

As mentioned above, the *OSTIS Standard* is the main part of the *OSTIS Metasystem* knowledge base describing the current version of the *OSTIS Technology*, as shown in Figure 1.

The proposed representation of the *OSTIS Standard* provides effective semantic navigation through the contents of the *OSTIS Standard*, since by going to the corresponding section of the *OSTIS Metasystem*, as shown in Figure 2, it is possible to see the current version of the *OSTIS Standard*.

The user is given the opportunity to go to any topic of interest to them (Figure 3) and ask the *OSTIS Metasystem* a wide range of non-trivial questions about the most diverse details and subtleties of the *OSTIS Technology*, as shown in Figure 4, and get answers to the questions asked, as shown in Figure 5.

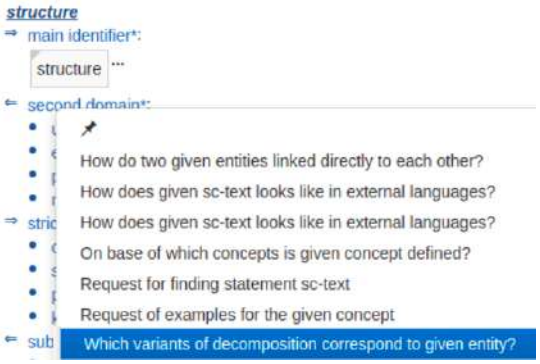


Subject domain of structures
⇒ main identifier*:
Subject domain of structures ...
→ primary class elements private subject domain*:
Subject domain of knowledges
⇐ primary class elements private subject domain*:
Subject domain of sc-elements
∈ key sc-element:
Subject domain and ontology of structures
∋ maximum class of explored objects':
structure
∋ not maximum class of explored objects':
- connected structure
- unconnected structure
- trivial structure
- nontrivial structure
- second structure element
- semantic level of structural element
- quantity of semantic level of structural element

∋ explored_relation':
- structure element'
- unrepresented set'
- completely represented set'
- partly represented set'
- structure element that are not set'
- maximum set'
- not maximum set'
- primary element'
- secondary element'
- second level element'
- metaconnection'
- polymorphic*

Figure 3. Navigation through the OSTIS Standard Table of contents

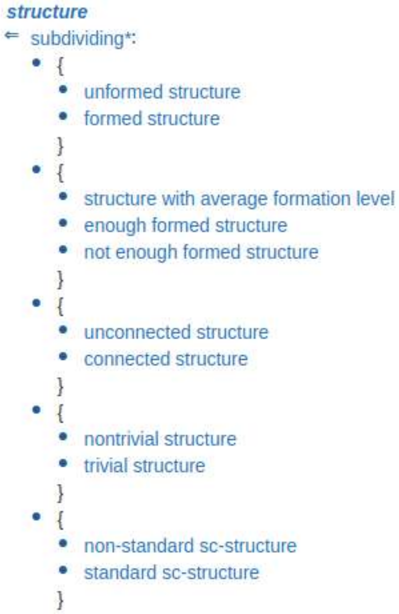
By default, the system responses to the user are displayed in the *SCn-code*, which is a hypertext version of the external display of *SC-code* texts and can be read as linear text.



structure
⇒ main identifier*:
structure ...
⇐ second domain*:
⇒ stric
⇐ sub

How do two given entities linked directly to each other?
How does given sc-text looks like in external languages?
How does given sc-text looks like in external languages?
On base of which concepts is given concept defined?
Request for finding statement sc-text
Request of examples for the given concept
Which variants of decomposition correspond to given entity?

Figure 4. The function of questions of the OSTIS Metasystem



structure
⇐ subdividing*:
- {
  - unformed structure
  - formed structure

  }
- {
  - structure with average formation level
  - enough formed structure
  - not enough formed structure

  }
- {
  - unconnected structure
  - connected structure

  }
- {
  - nontrivial structure
  - trivial structure

  }
- {
  - non-standard sc-structure
  - standard sc-structure

  }

Figure 5. The function of responses of the OSTIS Metasystem

XX. Conclusion

Semantic compatibility of intelligent computer systems is necessary for the implementation of cooperative, purposeful, and adaptive interaction of intelligent computer systems within automatically formed collectives of intelligent computer systems, and this, in turn, requires the unification of intelligent computer systems. Unification of an intelligent computer system is possible only on the basis of a general formal theory of intelligent computer systems and the corresponding *standard of intelligent computer systems*, but for this a deep convergence of various research directions in the field of Artificial Intelligence is necessary.

Since the result of developing Artificial Intelligence as a scientific discipline is the permanent evolution of the general theory of intelligent computer systems and the corresponding standard of intelligent computer systems, in order to increase the pace of development of Artificial

Intelligence and, accordingly, technology for the development of intelligent computer systems, it is necessary to create a portal of scientific and technical knowledge on Artificial Intelligence, ensuring the coordination of the activities of specialists, as well as the coordination and integration of the results of this activity.

In the article, an approach to automating the processes of creation, development, and application of standards based on the *OSTIS Technology* is considered. Based on the *Standard of the OSTIS Technology*, the basic principles underlying the proposed approach to standardization are considered.

The approach proposed in the work allows providing not only the possibility of automating the processes of creating, approving, and developing standards but also significantly increasing the efficiency of the processes for applying the standard, both manually and automatically.

Acknowledgment

The author would like to thank the research group of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for its help in the work and valuable comments.

References

[1] V. Golenkov, N. Guliakina, I. Davydenko, and A. Eremeev, "Methods and tools for ensuring compatibility of computer systems," *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, pp. 25–52, 2019.

[2] P. Serenkov, V. Solomaho, V. Nifagin, and A. Minova, "Koncepcija infrastruktury standartizacii kak bazy znanij na osnove ontologij [the concept of a standardization infrastructure as an ontology-based knowledge base]," *Novosti. Standartizacija i sertifikacija. [News. Standardization and certification.]*, 2004.

[3] V. Uglev, "Aktualizacija soderzhanija standartov proektirovanija slozhnyh tehnicheskih ob'ektov: ontologicheskij podhod [updating the content of design standards for complex technical objects: ontologic approach]," *Ontologija proektirovanija. [Ontology of designing]*, 2012.

[4] (2022, Nov) It/apkit professional standards. [Online]. Available: http://www.apkit.webtm.ru/committees/education/meetings/standarts.php

[5] A. I. Volkov, L. A. Reingold, and E. A. Reingold, "Profes-sional'nye standarty v oblasti it kak faktor tekhnologicheskogo i sotsial'nogo razvitiya [professional standards in the field of it as a factor of technological and social development]," *Prikladnaya informatika [Journal of applied informatics]*, pp. 80–86, 2015.

[6] I. Davydenko, "Semantic models, method and tools of knowledge bases coordinated development based on reusable components," *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, pp. 99–118, 2018.

[7] S. El-Sappagh, F. Franda, F. Ali, and K.-S. Kwak, "Nomed ct standard ontology based on the ontology for general medical science," *BMC Medical Informatics and Decision Making*, vol. 18, no. 1, 2018. [Online]. Available: https://doi.org/10.1186/s12911-018-0651-5

[8] B. R. Heravi, M. Lycett, and S. de Cesare, "Ontology- based standards development: Application of ontostand to ebxml business process specification schema," *International Journal of Accounting Information Systems*, vol. 15, no. 3, pp. 275–297, 2014. [Online]. Available: https://doi.org/10.1016/j.accinf.2014.01.005

[9] (2022, Nov) IMS.ostis Metasystem. [Online]. Available: https://ims.ostis.net

[10] V. Golenkov, N. Gulyakina, and D. Shunkevich, *Otkrytaja tehnologija ontologicheskogo proektirovanija, proizvodstva i jekspluatacii semanticheski sovmestimyh gibridnyh intellektual'nyh komp'juternyh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems].* Bestprint [Bestprint], 2021.

Метасистема Технологии OSTIS и Стандарт Технологии OSTIS

Банцевич К.А.

В данной работе предлагается подход к автоматизации процессов создания, развития и применения стандартов на основе Технологии OSTIS. Рассмотрены общие проблемы, связанные с развитием и применением современных стандартов в различных областях. Предложена стандартизация интеллектуальных компьютерных систем, а также стандартизация методов и средств их проектирования в рамках предлагаемого подхода.

Received 14.11.2022

Integration of computer algebra tools into OSTIS applications

Valery B. Taranchuk
Department of Computer Applications and Systems
Belarusian State University
Minsk, Republic of Belarus
taranchuk@bsu.by

Abstract—**From the standpoint of the need for convergence and unification of intelligent computer systems of a new generation, the issues of technology development and modernization, integration of the OSTIS metasystem with the Wolfram Mathematica computer algebra system are discussed. The current results and plans for the semantic representation of abstract mathematics on the Wolfram Language platform are noted as benchmarks. The software solutions implemented in the Wolfram Knowledgebase are marked and illustrated with examples.**

Index Terms—**Semantic analysis, Wolfram Mathematica, Wolfram Knowledgebase, Entity**

I. Introduction

Following the assessment given in [1] of the current state of work in the field of Artificial Intelligence (AI), we can note the active local development of various areas (non-classical logic, formal ontologies, artificial neural networks, machine learning, soft computing, multi-agent systems, etc.), but there is no comprehensive increase in the level of intelligence of modern intelligent computer systems. What is especially relevant at the moment? Convergence and integration of all areas of Artificial Intelligence and the corresponding construction of a general formal theory of intelligent computer systems (ICS) are required.

It is important to transform the modern variety of tools (frameworks) for the development of various components of ICS into a single technology for integrated design and support of the full life cycle of these systems, which guarantees the compatibility of all components being developed, as well as the compatibility of ICS themselves as independent entities interacting with each other within complex automation systems of complex types of collective human activity. Convergence and unification of intelligent computer systems of the new generation and their components is necessary. At the same time, convergent solutions mainly mean optimized complexes containing everything necessary to solve AI problems, organized or configured for the effective use of information resources, to simplify implementation processes; in particular, it should be possible to solve certain tasks with optimization requirements and achieve maximum performance, and in all implementations – optimized for ease of use. The key reasons of methodological problems of the current state of Artificial Intelligence, as well as a number of required actions to solve them are indicated in [1]. Supporting these concepts, we note that such problems are solved when developing, improving, systematically updating the content and expanding the capabilities of computer algebra systems. The following examples illustrate several methodological and technical solutions for convergence and integration of various types of knowledge implemented in the computer algebra system Wolfram Mathematica (WM), Wolfram Language (WL).

II. Wolfram Mathematica. The semantic representation of pure mathematics. The current state, plans

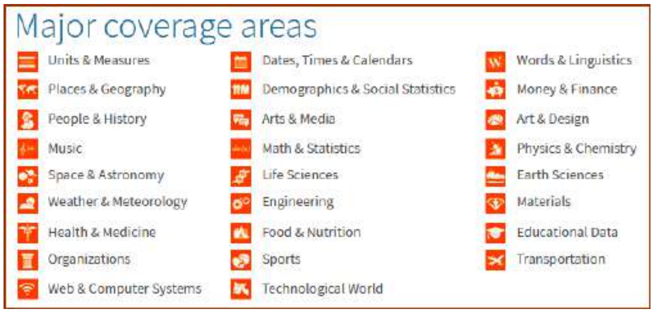
The current state. Based on more than thirty years of research, development and use throughout the world, Mathematica and the Wolfram Language are aimed at a long-term perspective and are particularly successful in computational mathematics. About 6000 characters embedded in the Wolfram Language allow you to represent and manipulate a huge variety of computational objects in the system – from special functions to graphics and geometric areas. In addition, the Wolfram Knowledgebase [2] and the associated entity structure [3] allow you to explain/interpret/formalize hundreds of specific "things (facts/situations/objects)". For example: people, cities, food, structures, planets, ... are represented by objects that can be manipulated, they can be cheated.

Wolfram Mathematica. Immediate plans. Despite a rapidly and ever-increasing number of domains known to WL, many knowledge domains still await computational representation. In his blog "Computational Knowledge and the Future of Pure Mathematics" Stephen Wolfram presented a grand vision for the representation of abstract mathematics, known variously as the Computable Archive of Mathematics or Mathematics Heritage Project (MHP). The eventual goal of this project is no less than to render all of the approximately 100 million pages of peer-reviewed research mathematics published over the last several centuries into a computer-readable form.

Wolfram Mathematica. The semantic representation of pure mathematics. In the blog [4], leading Wolfram specialists present their vision of the future semantic representation of abstract mathematics using two examples: abstract mathematical concepts of functional, topological spaces; concepts and theorems of general topology. It seems that such concepts and approaches should be used in solving methodological problems of the current state of Artificial Intelligence.

III. The Wolfram Knowledgebase. Examples

Powering Wolfram|Alpha and WL, the ever-growing Wolfram Knowledgebase (WKB) is by far the world's largest and broadest repository of computable knowledge. WKB, covering thousands of fields, contains carefully selected expert knowledge directly derived from primary sources ([4]). It includes not only trillions of data elements, but also immense number of algorithms encapsulating methods and models from almost every field. The main subject fields of WKB are illustrated in Fig. 1.



Major coverage areas

Units & Measures
Places & Geography
People & History
Music
Space & Astronomy
Weather & Meteorology
Health & Medicine
Organizations
Web & Computer Systems

Dates, Times & Calendars
Demographics & Social Statistics
Arts & Media
Math & Statistics
Life Sciences
Engineering
Food & Nutrition
Sports
Technological World

Words & Linguistics
Money & Finance
Art & Design
Physics & Chemistry
Earth Sciences
Materials
Educational Data
Transportation

Figure 1. The main subject fields of WKB.

The Wolfram Knowledgebase relies on three decades of computable knowledge acquisition. All data in the Wolfram database can immediately be used for computation in WL. Every millisecond of every day, WKB is updated with the latest data.

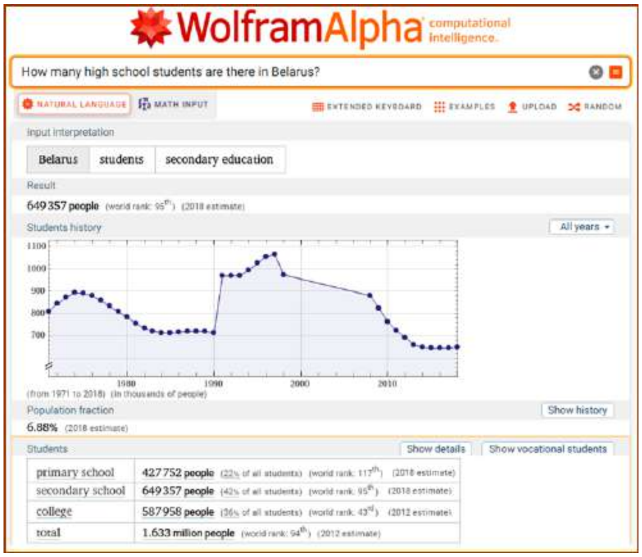
Let's note a few examples. With a trove of statistics for hundreds of thousands of educational institutions around the world, Wolfram|Alpha can compute answers to intricate questions about education You can request which academic degrees students of prestigious universities receive. You can also compute the average salary for teachers in your local school district, learn more about student scores, and compare student-teacher ratios among countries and much more. Fig. 2 illustrates the response to the request for the number of students in the Republic of Belarus.

The comparison for the universities of BSU and BSUIR is illustrated in Fig. 3.

IV. The Wolfram Knowledgebase. Representation and access to it

Access to WKB is deeply integrated into WL. Free-form linguistics makes it easy to identify many millions of entities and many thousands of properties and automatically generate precise WL representations suitable for extensive further computation. The Wolfram Language also supports custom entity stores that allow the same computations as the built-in knowledgebase, and can be associated with external relational databases. Note the main groups of WM functions for working with WKB: Entity & EntityClass & EntityValue, Transformations & Computations on Entity Classes, Standard Properties, Specific Domains, Setting Up Custom Entity Stores, Wolfram Data Repository, Wolfram Data Drop, Setting Up Custom Entity Stores, External Knowledgebases, External Database Connectivity, Web Content, Textual Question Answering, System Configuration. There are more than three subgroups in each of the listed groups. For example, the Textual Question Answering group includes:

- *FindTextualAnswer* attempt to find answers to questions from text;
- *SemanticInterpretation* convert free-form linguistics to Wolfram Language for; *SemanticInterpretation*["string"] attempts to give the best semantic interpretation of the specified free-form string as a Wolfram Language expression;
- *SemanticImport* import data, converting entities etc. to Wolfram Language form,
- *Interpreter* interpret input of various types (e.g. "City", "Date", etc.); Interpreter attempt to interpret strings of a wide variety of types; *Interpreter*[form] represents an interpreter



How many high school students are there in Belarus?

Input interpretation

Belarus	students	secondary education

Result

649 357 people (world rank: 95th) (2018 estimate)

Students history

(from 1971 to 2018) (in thousands of people)

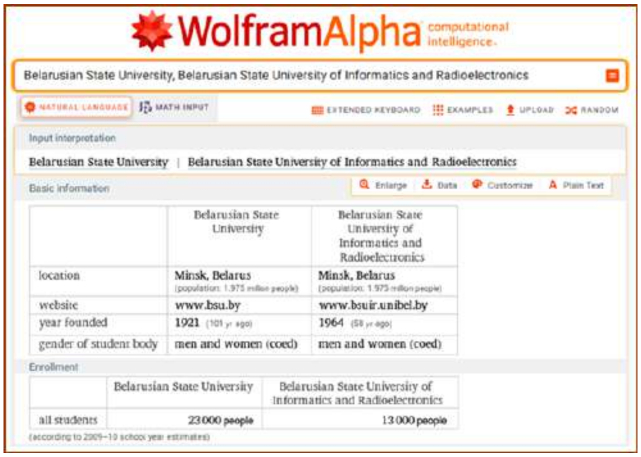
Population fraction

6.88% (2018 estimate)

Students

primary school	427 752 people (22% of all students) (world rank: 117th) (2018 estimate)
secondary school	649 357 people (42% of all students) (world rank: 95th) (2018 estimate)
college	587 958 people (36% of all students) (world rank: 43rd) (2012 estimate)
total	1.633 million people (world rank: 94th) (2012 estimate)

Figure 2. How many high school students are there in Belarus.



Belarusian State University, Belarusian State University of Informatics and Radioelectronics

Input interpretation

Belarusian State University | Belarusian State University of Informatics and Radioelectronics

Basic information

	Belarusian State University	Belarusian State University of Informatics and Radioelectronics
location	Minsk, Belarus (population: 1.975 million people)	Minsk, Belarus (population: 1.975 million people)
website	www.bsu.by	www.bsuir.unibel.by
year founded	1921 (101 yr ago)	1964 (58 yr ago)
gender of student body	men and women (coed)	men and women (coed)

Enrollment

	Belarusian State University	Belarusian State University of Informatics and Radioelectronics
all students	23 000 people	13 000 people

(according to 2009–10 school year estimates)

Figure 3. How many high school students are there in Belarus.

object that can be applied to an input to try to interpret it as an object of the specified form.

V. Semantic Analysis

Humans interact with each other through speech and text, and this is called Natural language. Computers understand the natural language of humans through Natural Language Processing (NLP). NLP is a process of manipulating the speech of text by humans through Artificial Intelligence so that computers can understand them. Human language has many meanings beyond the literal meaning of the words. There are many words that have different meanings, or any sentence can have different tones like emotional or sarcastic. It is very hard for computers to interpret the meaning of those sentences.

NLP. Main applications, tools implemented in the Wolfram Mathematica system: Speech Recognition, Voice Assistants and Chatbots, Auto Correct and Auto prediction, Email Filtering, Sentiment Analysis, Divertissements to Targeted Audience, Translation, Social Media Analytics, Recruitment, Text Summarisation.

Several representative examples with explanations of the functions of the WL groups Structural Text Manipulation, Text Analysis, Natural Language Processing are mentioned below. In a sense, these categories are conditional, there are a lot of functions and capabilities implemented by them. For example, the Structural Text Manipulation subgroup may include the following: TextCases – extract symbolically specified elements (*TextCases*[text,form] gives a list of all cases of text identified as being of type form that appear in text); TextSentences – extract a list of sentences (*TextSentences*["string"] gives a list of the runs of characters identified as sentences in string); TextWords – extract a list of words (*TextWords*["string"] gives a list of the runs of characters identified as words in string); SequenceAlignment – find matching sequences in text; *TextStructure*["text"] generates a nested collection of TextElement objects representing the grammatical structure of natural language text [5].

An example and the result of executing the TextStructure function to the text "Open Semantic Technologies for Intelligent Systems" with the option "ConstituentTree" is shown in Fig. 4

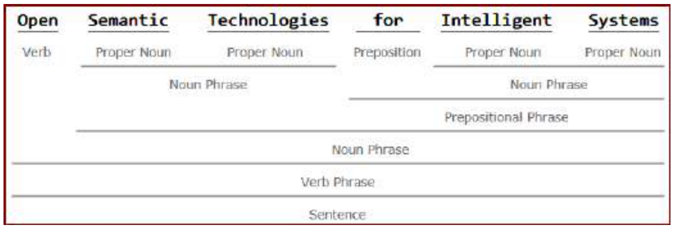


Figure 4. The result of executing the TextStructure function with the option "ConstituentTree".

Variants for visualizing the components of the analyzed phrase with the settings "ConstituentGraphs", "DependencyGraphs" are shown in Fig. 5.

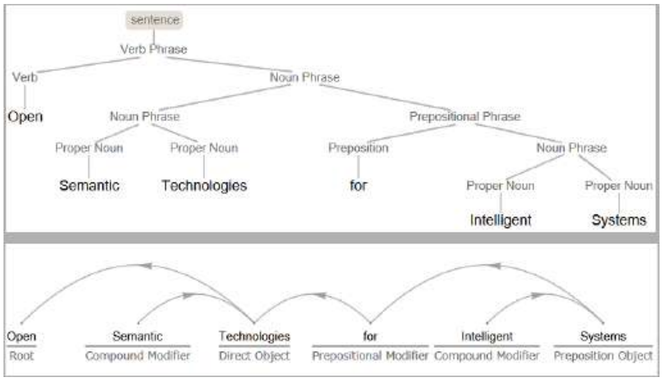


Figure 5. Results of executing the TextStructure function with the settings "ConstituentGraphs", "DependencyGraphs".

NLP. Examples of using the FindTextualAnswer function. Answer questions in natural language from the text [6].

FindTextualAnswer[text,"question",n] gives a list of up to n answers that appear most probable. *FindTextualAnswer*[text,"question",n,prop] gives the specified property for each answer.

In the two examples below, the processing object is text *International scientific and technical conference proceedings "Open Semantic Technologies for Intelligent Systems (OSTIS)". Established: 2011. Scientific areas of the conference: 05.13.11, 05.13.15, 05.13.17*. In the query variant with the option "Date of establishment of the conference?" the answer is

"2011"

In the request variant with the options "Date of the conference establishment?", "Scientific directions of the conference?" the answer is a list

"2011", "05.13.11, 05.13.15, 05.13.17"

In the following example, the processing object is text *International scientific and technical conference proceedings Open Semantic Technologies for Intelligent Systems (OSTIS). Established: 2011. Scientific areas of the conference: Theory of Informatics, Software for Computers, Computer Complexes and Networks, Computing Machines and Complexes and Computer Networks*.

In the query variant with the option "Scientific areas of the conference?" the answer is –

"Theory of Informatics, Software for Computers, Computer Complexes and Networks, Computing Machines and Complexes and Computer Networks" .

Next example illustrates the search in the text and the fixation of three signs. The object of processing is text *International scientific and technical conference proceedings Open Semantic Technologies for Intelligent Systems (OSTIS). Established: 2011. Program Committee: Kuznetsov Oleg Co-Chair, Dr. of Techn. Sciences, Professor, Academician of the Russian*

Academy of Natural Sciences, Moscow, Russia; Golenkov Vladimir Co-Chair, Dr. of Techn. Sciences, Professor, Minsk, Belarus; ... Arefiev Igor Dr. of Techn. Sciences, Professor, Szczecin, Poland; ... Globa Larisa Dr. of Techn. Sciences, Professor, Kyiv, Ukraine ... .

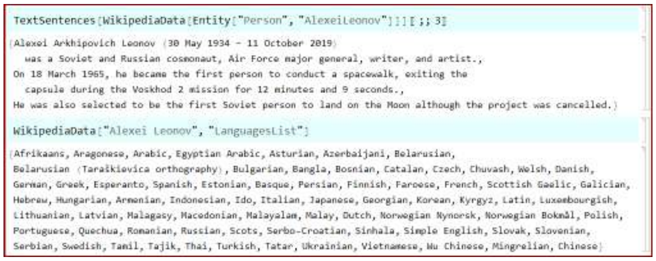
In the query variant with the options "City", "Country", "Date", the response is –

```
<|"City" -> "Moscow", "Minsk", "Szczecin", "Kyiv",
"Country" -> "Russian", "Russia", "Belarus", "Poland",
"Ukraine", "Date" -> "2011"|>
```

.

VI. Examples of extracting knowledge, entities, or topics from Wikipedia articles

WikipediaData utilizes MediaWiki's API to retrieve article and category contents and metadata from Wikipedia [7]. An article may be specified as a string or a Wolfram Language entity. The extraction of articles associated with language entities is provided by the WM TextSentences function, in particular, you can work with Wikipedia resources. *TextSentences*["string"] gives a list of the runs of characters identified as sentences in string [8]. *WikipediaData*[article] gives the plain text of the specified Wikipedia article. *Entity*["type",name] represents an entity of the specified type, identified by name. *WikipediaData*[article,property,options] gives the value of the specified property, modified by optional parameters, for the given Wikipedia article. Below are the results of executing the *TextSentence* function, with the parameters WikipediaData, Entity, "Person", "AlexeiLeonov" and displaying a list of language versions of Wikipedia (*LanguagesList*) containing the corresponding article (in Fig. 6)



```
TextSentences[WikipediaData[Entity["Person", "AlexeiLeonov"]]][[;; 3]]
{Alexei Arkhipovich Leonov (30 May 1934 – 11 October 2019)
  was a Soviet and Russian cosmonaut, Air Force major general, writer, and artist.,
On 18 March 1965, he became the first person to conduct a spacewalk, exiting the
  capsule during the Voskhod 2 mission for 12 minutes and 9 seconds.,
He was also selected to be the first Soviet person to land on the Moon although the project was cancelled.}

WikipediaData["Alexei Leonov", "LanguagesList"]
{Afrikaans, Aragonese, Arabic, Egyptian Arabic, Asturian, Azerbaijani, Belarusian,
Belarusian (Taraškievica orthography), Bulgarian, Bangla, Bosnian, Catalan, Czech, Chuvash, Welsh, Danish,
German, Greek, Esperanto, Spanish, Estonian, Basque, Persian, Finnish, Faroese, French, Scottish Gaelic, Galician,
Hebrew, Hungarian, Armenian, Indonesian, Ido, Italian, Japanese, Georgian, Korean, Kyrgyz, Latin, Luxembourgish,
Lithuanian, Latvian, Malagasy, Macedonian, Malayalam, Malay, Dutch, Norwegian Nynorsk, Norwegian Bokmål, Polish,
Portuguese, Quechua, Romanian, Russian, Scots, Serbo-Croatian, Sinhala, Simple English, Slovak, Slovenian,
Serbian, Swedish, Tamil, Tajik, Thai, Turkish, Tatar, Ukrainian, Vietnamese, Wu Chinese, Mingrelian, Chinese}
```

Figure 6. TextSentences. WikipediaData. Entity. AlexeiLeonov. LanguagesList.

Fig. 7 illustrates the system's response to the execution of the WikipediaData function with the parameters "Voskhod 2", "ImageList":

List of rules representing links between categories. *WikipediaData*["Category"->category,property,options] gives the value of the specified property, modified by optional parameters, for the given Wikipedia category. "MaxLevelItems" – number of links to follow at each level. "MaxLevel" – number of levels to search outward from the specified page.

Fig. 8 shows the result of the function *WikipediaData*["Category"->"Artificial intelligence", "CategoryLinks", "MaxLevelItems"->5, "MaxLevel"->3] execution in the form of a graph:

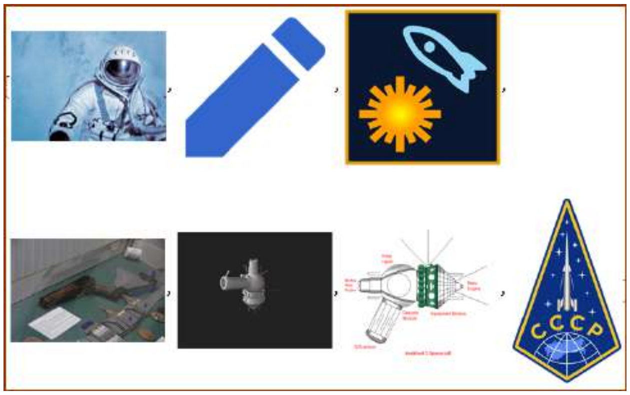
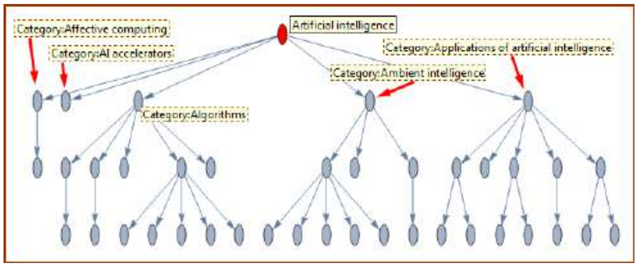


Figure 7. TextSentences. WikipediaData. Entity. Voskhod 2. ImageList.

You can output all categories separately, for example, for Applications of artificial intelligence, the system will output: "Category:Agent-based software", "Category:Applied data mining", "Category:Applied machine learning", "Category:Automated planning and scheduling", "Category:Computer vision software".

Examples of **extracting in the knowledge cloud, entities or topics, lists of rules representing relationships between categories** are given in the description of the WordCloud function. (*WordCloud*[s_1,s_2,...] generates a word cloud graphic in which the s_i are sized according to their multiplicity in the list.) The illustration in Fig. 9 is obtained in Wolfram Mathematica, the functions Delete stop words and Text Words are used. (*DeleteStopwords*[list] deletes stopwords from a list of words; *TextWords*["string"] gives a list of the runs of characters identified as words in string).

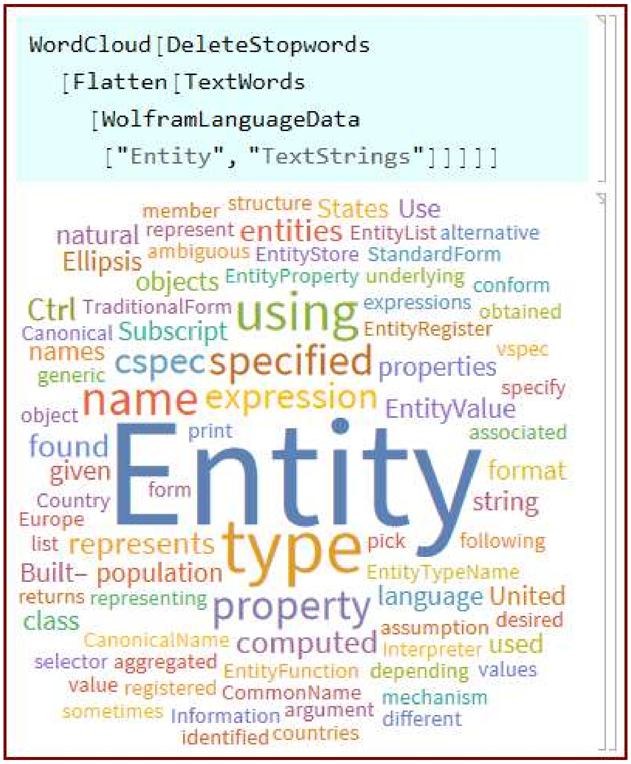


Category:Affective computing
Category:AI accelerators
Artificial intelligence
Category:Applications of artificial intelligence
Category:Ambient intelligence
Category:Algorithms

Figure 8. Graph for "Artificial intelligence".

VII. Intellectualization of user interfaces. Examples of implementation in Wolfram Mathematica

Currently, the user interfaces of many computer systems (including intelligent computer systems) in most implementations are not semantically friendly. For users, interaction with computer systems is often a "bottleneck" that has a significant impact on the efficiency of automation of human activities. The basis of the modern organization of user interaction with a computer system is the paradigm of a trained, competent user



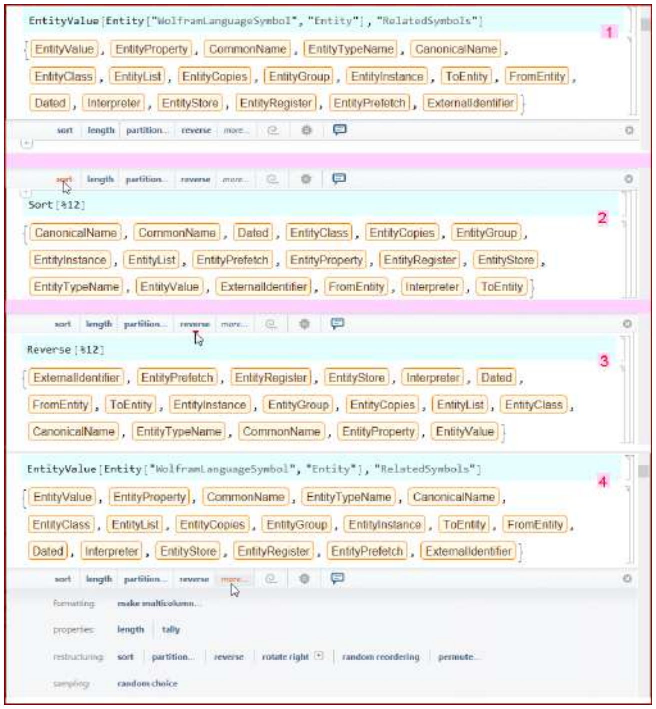
```
WordCloud[DeleteStopwords
  [Flatten[TextWords
    [WolframLanguageData
     ["Entity", "TextStrings"]]]]]
```

member structure States Use
natural represent entities EntityList alternative
Ellipsis ambiguous EntityStore StandardForm
objects EntityProperty underlying conform
Ctrl TraditionalForm using expressions obtained
Canonical Subscript EntityRegister
names vspec
generic cspec specified properties
specify
object name expression EntityValue
print associated
found Entity format
given form string
Country
Europe
list represents type pick following
Built– population EntityTypeName
returns representing property language United
class assumption desired
CanonicalName computed Interpreter used
selector aggregated EntityFunction depending values
value registered CommonName mechanism
sometimes Information argument different
identified countries

Figure 9. WordCloud for "Entity".

who knows the capabilities of the tool he uses, is responsible for the correctness of interaction with him. At the present stage of the development of Artificial intelligence, in order to increase the efficiency of interaction, it is necessary to move from the paradigm of competent management of the tool used to the paradigm of equal cooperation, partnership interaction of an intelligent computer system with the user. Semantic friendliness of the user interface should consist in adaptability to the features and qualifications of the user, eliminating problems for the user in the process of dialogue with the computer system. It is fundamental to switch from a friendly user interface to an intelligent predictive interface, in which the system, when working with a computer, not only speeds up the input of queries, simplifies the dialogue, clarifies the correctness of commands and actions, but also offers, after the output of the next result, a line of options for the following calculations, actions. Moreover, for each user, such a line of hints (prompts) is formed taking into account the accumulated statistics of individual preliminary requests and the content of the system, and in each particular session it includes links to several possible formalization approaches, several different scenarios of work on examples of similar tasks, virtual textbooks. According to the accumulated knowledge in the navigation block of the line of hints, the user can go to illustrations of other recommended processing methods by means of computer mathematics (with reference to the subject area being performed), interpretation algorithms.

Explanations, several typical illustrations, fragments of the windows of the user interface of the Mathematica system, in which the intelligent predictive interface is implemented, are shown in the following figures 10, 12.



```
EntityValue[Entity["WolframLanguageSymbol","Entity"],"RelatedSymbols"]
{EntityValue, EntityProperty, CommonName, EntityTypeName, CanonicalName,
EntityClass, EntityList, EntityCopies, EntityGroup, EntityInstance, ToEntity, FromEntity,
Dated, Interpreter, EntityStore, EntityRegister, EntityPrefetch, ExternalIdentifier}
```

```
Sort[%12]
{CanonicalName, CommonName, Dated, EntityClass, EntityCopies, EntityGroup,
EntityInstance, EntityList, EntityPrefetch, EntityProperty, EntityRegister, EntityStore,
EntityTypeName, EntityValue, ExternalIdentifier, FromEntity, Interpreter, ToEntity}
```

```
Reverse[%12]
{ExternalIdentifier, EntityPrefetch, EntityRegister, EntityStore, Interpreter, Dated,
FromEntity, ToEntity, EntityInstance, EntityGroup, EntityCopies, EntityList, EntityClass,
CanonicalName, EntityTypeName, CommonName, EntityProperty, EntityValue}
```

```
EntityValue[Entity["WolframLanguageSymbol","Entity"],"RelatedSymbols"]
{EntityValue, EntityProperty, CommonName, EntityTypeName, CanonicalName,
EntityClass, EntityList, EntityCopies, EntityGroup, EntityInstance, ToEntity, FromEntity,
Dated, Interpreter, EntityStore, EntityRegister, EntityPrefetch, ExternalIdentifier}
```

1
sort length partition... reverse more...
sort length partition... reverse more...
2
sort length partition... reverse more...
3
4
sort length partition... reverse more...
formatting: make multicolumn...
properties: length tally
restructuring: sort partition... reverse rotate right random reordering permute...
sampling: random choice

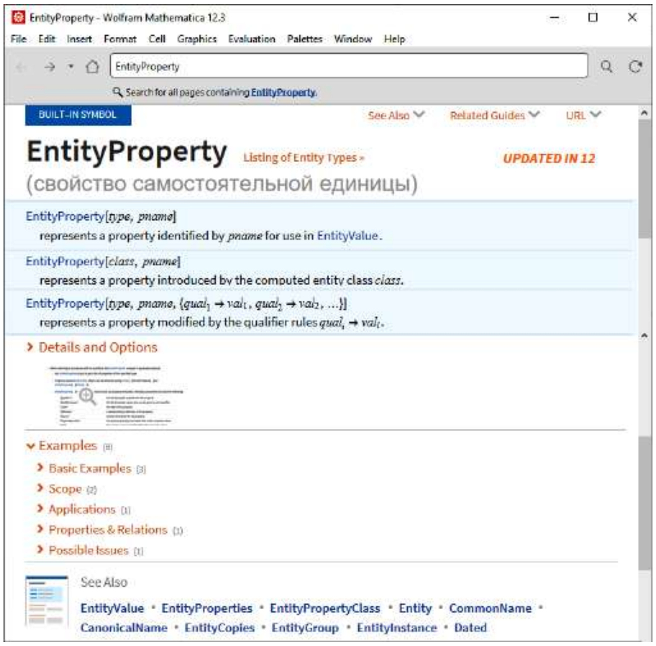
Figure 10. EntityValue. Entity. RelatedSymbols..

Each illustration includes several fragments of copies of the screens. Examples are given for the above request to display a cloud of documentation words for Entity, and for related WM functions associated to Entity [5], [6]. As a result of executing the section with the *EntityValue*[Entity["WolframLanguageSymbol", "Entity"], "RelatedSymbols"] command, the user receives an answer and a line of suggestions for further actions. Similar tooltips are displayed after each section is completed, and in all cases their content is determined by semantic connections. Fragment 2 Fig. 10 – the functions found and shown are sorted alphabetically, the contents of fragment 3 – the functions are listed in reverse order, fragment 4 contains an expanded menu of possible further actions.

All functions of the Wolfram Mathematica system are documented in detail; clicking on the function name in any section of the system provides access to a notebook-article – description. The examples given in the description can be executed in the notebook-article itself, you can copy and transfer to any other notebook, there are no restrictions on replacing data and settings.

Figure 11 shows a fragment of one (randomly selected) of the functions mentioned in the list – EntityProperty.

The illustration in Fig. 11, sections of the article-descriptions are typical; the structure is the same for everyone. It is important that the examples are grouped by difficulty levels, but there is no need to perform them sequentially.


EntityProperty - Wolfram Mathematica 12.3
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
EntityProperty
Search for all pages containing EntityProperty.
BUILT-IN SYMBOL
See Also
Related Guides
URL
Listing of Entity Types »
UPDATED IN 12

EntityProperty

(свойство самостоятельной единицы)

EntityProperty[*type*, *pname*]
represents a property identified by *pname* for use in EntityValue.

EntityProperty[*class*, *pname*]
represents a property introduced by the computed entity class *class*.

EntityProperty[*type*, *pname*, {$qual_1 \rightarrow val_1$, $qual_2 \rightarrow val_2$, …}]
represents a property modified by the qualifier rules $qual_i \rightarrow val_i$.

Details and Options
Examples (8)
Basic Examples (3)
Scope (2)
Applications (1)
Properties & Relations (1)
Possible Issues (1)
See Also
EntityValue • EntityProperties • EntityPropertyClass • Entity • CommonName •
CanonicalName • EntityCopies • EntityGroup • EntityInstance • Dated

Figure 11. EntityProperty. Article-description (notebook document) structure

Figure 12 shows the steps of work when the country (United States) is changed (Belarus) in the basic example, and then queries are selected following the prompts in the tooltips.

It should be noted that it is also possible to state certain successes in solving problems of underground hydrodynamics [9] , forest fires [10] on the formation and filling of knowledge bases with Wolfram Mathematica tools during processing, accumulation and interpretation of the results of computational experiments.

VIII. Conclusion

Several representative examples of working with knowledge bases by means of the Wolfram Mathematica system are discussed. Since the functions of the Mathematica core can be used in programs developed on other platforms, presented results can be interpreted as proposals for innovative improvement of existing tools, components of intelligent computer systems.

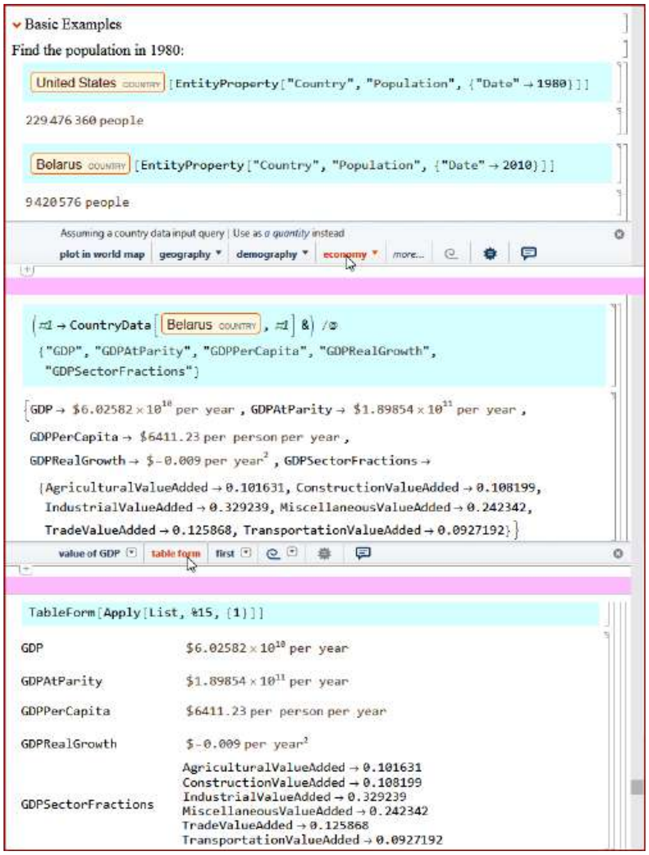
References

[1] V. Golenkov, N. Guliakina, and D. Shunkevich, Otkrytaja tehnologija ontologicheskogo proektirovanija, proizvodstva i jekspluatacii semanticheski sovmestimyh gibridnyh intellektual'nyh komp'juternyh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems], V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[2] Wolfram Knowledgebase. Making the knowledge of the world computable. Available at: https://www.wolfram.com/knowledgebase (accessed 2022, Oct).

[3] Knowledge Representation & Access. Wolfram Language & System Documentation Center. Available at: https://reference.wolfram.com/language/guide /KnowledgeRepresentationAndAccess.html (accessed 2022, Oct).

[4] The Semantic Representation of Pure Mathematics. Available at: https://blog.wolfram.com/2016/12/22/the-semantic-representation-of-pure-mathematics/ (accessed 2022, Oct).



Find the population in 1980:

```
United States COUNTRY [EntityProperty["Country", "Population", {"Date" → 1980}]]

229 476 360 people

Belarus COUNTRY [EntityProperty["Country", "Population", {"Date" → 2010}]]

9 420 576 people
```

Assuming a country data input query | Use as a quantity instead
plot in world map
geography
demography
economy
more...

```
(#1 → CountryData[Belarus COUNTRY, #1] &) /@
 {"GDP", "GDPAtParity", "GDPPerCapita", "GDPRealGrowth",
  "GDPSectorFractions"}

{GDP → $6.02582 × 10^10 per year , GDPAtParity → $1.89854 × 10^11 per year ,
 GDPPerCapita → $6411.23 per person per year ,
 GDPRealGrowth → $-0.009 per year^2 , GDPSectorFractions →
  {AgriculturalValueAdded → 0.101631, ConstructionValueAdded → 0.108199,
   IndustrialValueAdded → 0.329239, MiscellaneousValueAdded → 0.242342,
   TradeValueAdded → 0.125868, TransportationValueAdded → 0.0927192}}
```

```
TableForm[Apply[List, %15, {1}]]

GDP                   $6.02582 × 10^10 per year
GDPAtParity           $1.89854 × 10^11 per year
GDPPerCapita          $6411.23 per person per year
GDPRealGrowth         $-0.009 per year^2
                      AgriculturalValueAdded → 0.101631
                      ConstructionValueAdded → 0.108199
GDPSectorFractions    IndustrialValueAdded → 0.329239
                      MiscellaneousValueAdded → 0.242342
                      TradeValueAdded → 0.125868
                      TransportationValueAdded → 0.0927192
```

Figure 12. An illustration of the use of a ruler (palette) of suggestions for the following calculations.

[5] TextStructure. Available at: https://reference.wolfram.com/language/ref/TextStructure.html/ (accessed 2022, Oct).

[6] FindTextualAnswer. Available at: https://reference.wolfram.com/language/ref/FindTextualAnswer.html/ (accessed 2022, Oct).

[7] WikipediaData. Available at: https://reference.wolfram.com/language/ref/WikipediaData.html/ (accessed 2022, Oct).

[8] TextSentences. Available at: https://reference.wolfram.com/language/ref/TextSentences.html/ (accessed 2022, Oct).

[9] V. Taranchuk, Tools and examples of intelligent processing, visualization and interpretation of GEODATA, Modelling and Methods of Structural Analysis. IOP Conf. Series: Journal of Physics: Conf. Series Vol. 1425 (2020) 012160. – P. 9, doi:10.1088/1742-6596/1425/1/012160.

[10] D.V. Barovik, V.B Taranchuk. Tools for the analysis and visualisation of distributions and vector fields in surface forest fires modelling. Journal of the Belarusian State University. Mathematics and Informatics. – 2. – 2022. – P. 82-93.

Интеграция инструментов компьютерной алгебры в приложения OSTIS

Таранчук В. Б.

С позиций необходимости конвергенции и унификации интеллектуальных компьютерных систем нового поколения обсуждаются вопросы технологии разработки и модернизации, интеграции средств метасистемы OSTIS с системой компьютерной алгебры Wolfram Mathematica.

Received 15.11.2022

Semantically Compatible OSTIS Educational Automative Systems

Natalya Gulyakina
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
Email: guliakina@bsuir.by

Alena Kazlova
Belarusian State University
Minsk, Belarus
Email: kozlova@bsu.by

***Abstract*—A class of semantic electronic textbooks is proposed, which are based on the semantic structuring of educational material. Thanks to the semantic structuring of the educational and methodological material, the SEU acquires new opportunities compared to traditional electronic textbooks. A semantic electronic textbook is an interactive intellectual self-instruction manual for a certain subject area, containing detailed methodological recommendations for studying it and intended for a motivated, independent and active user who wants to acquire knowledge in the relevant discipline (subject area). The prospect of designing intelligent learning systems for specialties is considered on the example of the specialty Artificial Intelligence.**

***Keywords*—semantics, electronic textbooks, knowledge, hypertext semantic network, intelligent learning systems**

I. Introduction

The organization of educational activities today largely determines the level of development of any state and society. Therefore, we can fully explain the great interest in the use of telecommunications and computer technologies in order to increase the efficiency of this activity. In this regard, such a direction of research in the field of artificial intelligence as intelligent learning systems and automation of educational activities is of particular relevance. Many authors in our days consider the problems of transition to digital forms of education [1], [2], [3], [4].Intelligent learning systems (ILS) should become part of the specialist training complex. At the same time, such systems can be effectively used in the process of providing advanced training, in the implementation of continuous education. Also, one of the features of the intelligent learning systems development is that the users of such systems will be non-specialists in the field of computer and telecommunication technologies, people who are significantly different both in age and in the level of knowledge in a particular subject area. This is an incentive for the development of artificial intelligence technologies in various applied areas of human activity in order to realize the possibility of building an ILS for training specialists in various professional areas. The most promising from the point of view of the development and implementation of ILS in the educational process of universities is the use of the OSTIS ecosystem for the formation of both elements of learning and educational systems, and their integration into a single complex.

II. 1. Semantic electronic textbook as a new type of computer learning tools

Along with traditional sources of knowledge, such as teachers (lecturers and seminar leaders), books (teaching aids), family, acquaintances, the members of the study group, etc., today such sources, depersonalized in the sense of personal communication, as intellectual (computer) information systems, specialized data- and knowledge bases, electronic textbooks, including those prepared using hypermedia and multimedia tools, as well as Internet-based network sources. Thus, the ever-increasing requirements for the efficiency and practical orientation of training systems lead to the inevitable realization of the relevance of the problem of developing such computer training systems, that provide:

1) processing of large volumes of complexly structured information of various types;
2) flexibility and easy modifiability of the system;
3) integration of various models and mechanisms for solving problems;
4) support for various models of learning and user interaction management;
5) integration of various software systems within one system and management of their operation and interaction;
6) wide use of multimedia tools;
7) operation in real time.

An example of building such intelligent learning systems can be ILS based on semantic electronic textbooks (SET). An electronic textbook, as a rule, is used for independent study of an academic discipline. As far as the knowledge presented in the electronic textbook is well structured, their completeness and consistency, clarity and accessibility, the possibility of quick associative search are ensured, the effectiveness of the learning process increases. Semantic electronic textbook – a set of software tools built using methods and tools of artificial intelligence, in particular, OSTIS technology, in which the knowledge base of educational and educational material is presented in the form of a hypertext semantic network and the possibility of associative access to any fragment of this educational material is provided. Feature of the semantic electronic textbook in comparison with the traditional electronic textbook:

1) SET is a software system designed both for the development of electronic textbooks of this type, and for the user (student) to work with this textbook.
2) The SET is based on a knowledge base, which integrates the formal presentation of educational material with ILS traditional hypertext and hypermedia presentation, while maintaining all the positive features of the latter.
3) Since the SET is based on the knowledge base, the electronic textbook turns into a fairly "reasonable" question-answer system for the specified material (i.e., into a system capable of finding answers to a fairly large number of questions related to the meaning, semantics of the relevant material). This means that the SET becomes a system that "understands"the meaning of the educational and teaching materials contained in it.
4) The SET provides the possibility of visualization (including three-dimensional) of the semantic structures of the educational material, presented in the form of semantic networks.

The formal basis for the representation of knowledge in the SET is hypertext semantic networks (HSN) [5]. The composition of the semantic electronic textbook, in addition to the actual educational material, includes:

1) Knowledge base of the subject area, which is a formal record of the semantics of educational material in the knowledge representation language, which also includes:
   - a set of links between the generated knowledge base and information sources for this knowledge base;
   - a description of the specifications of the fragments of educational material presented in one form or another (information about what software was used to develop this fragment, how this fragment is related to other fragments of educational material, etc.);
   - various kinds of systematization, structuring and meta-description of educational material.
2) Subsystem for the formation and editing of educational material. This subsystem is designed to acquire expert knowledge. The formation of the knowledge base takes place in the representation language of hypertext semantic networks (SCht) and ILS extension, specially oriented to describe educational material (both in linear and graph language specifications). This subsystem supports the possibility of further checking the syntactic and semantic correctness of the generated knowledge base.
3) Subsystem of navigation through educational material. As a language of dialogue with the user, a specialized query language or menu items are used.

The semantic electronic textbook is generally focused on working with such categories of users as the authors of educational material (subject expert, expert teacher, knowledge engineer, designer) and students. As practice shows, often one person often performs the functions of all the listed authors (developers) of an electronic textbook.

The SET operates in two modes: in the mode of formation of educational material (the mode of acquiring knowledge) and in the training mode (presenting the material to the end user-learner). The user interface of the SET includes:

- commands for the formation of the educational material of the SET;
- commands for editing the educational material of the SET;
- commands for navigation through the hypertext semantic network of the SET;
- the command for printing the hypertext semantic network of the SET;
- teams that ensure the integration of the SET;
- graphical modification of the universal language of semantic networks SC;
- linear modification of the universal language of semantic networks SC;
- ways to visualize various other forms of presentation of educational material (editors of traditional information structures).

III. Hypertext Semantic Networks as a Model of the SEU Subject Domain

The domain model is used to solve the problems of structuring and systematizing educational material, implementing navigation and search algorithms for educational material, generating information about the student and implementing adaptive learning management, etc. The knowledge representation model should combine traditional forms of presenting educational information with ILS formal presentation. In this paper, it is proposed to use hypertext semantic networks as a knowledge representation model [5]. Hypertext semantic networks (HSNs) are a class of knowledge representation models, the distinguishing feature of which is that various forms of information representation are integrated on their basis: traditional, for example, hypermedia, and formal – in the form of semantic networks, in fact, the knowledge base. Traditional forms of information presentation are focused primarily on the visualization of the information displayed to the user, in turn, the formalized presentation of this information in the knowledge representation language makes it a semantically interpreted system. The integration of these heterogeneous ways of presenting information is based on meta-relations (meta-descriptions) of this information. HSN is a knowledge representation model focused on a formal description of the syntax and semantics of information structures of any kind, as well as a description of the relationships between them.

A hypertext semantic web can also be called a semantically structured hypertext; the result of the integration of hypertext technologies and technologies based on semantic networks; semantically structured hypertext multimedia knowledge base. The following should be indicated as the main structures underlying hypertext semantic networks. The HSN node, which is a sign of some information structure, contains the information structure designated by it (the information structure is considered to be the content of the node of the hypertext semantic network that designates it). The set of input signs denoting a variety of objects (specific objects of a certain subject area, specific information structures, specific sets, connections, concepts, relations) of the HSN one-to-one correspond to a set of identifiers (names), which are a string (linear-symbolic) version of the image of signs. A HSN node that denotes some information construct does not always need to explicitly "store"that information construct as ILS content. This information structure may be unknown (not formed), it may be broken into fragments, each of which is presented explicitly, and, therefore, there is no need to explicitly represent and store the entire original information structure. An information structure (in particular, a text information structure) may include identifiers (names) of some characters represented by GSS nodes. This is interpreted as a link to the corresponding sign (a node of the hypertext semantic network). Links can be multisets, since the same element can be included in the link multiple times, including under different attributes. A link whose elements have their roles (attributes) is a directed link, otherwise it is an undirected link. Hypertext semantic networks allow you to have links that connect not only objects, information structures, but also the links themselves. They also allow the formation of set signs, called set systems, including the signs of some other sets and their elements, as well as the corresponding signs of membership pairs. In this way, it is possible to describe connections not only between bundles, but also between entire structures (systems of sets). GSN are focused on the description of subject knowledge that has a complex hierarchical, multi-level structure. In hypertext semantic networks, the description of various types of links is supported: links between links, links between entire structures, links between various fragments of processed knowledge. The set of nodes of the GSN on the subject basis is divided into a set of nodes of the subject level, that is, those that can be semantically interpreted directly through the elements, structures and relationships of the described subject area, and the set of meta-level nodes denoting statements about objects and relations of the subject area, the connections between them. These structures allow for semantic and semantic compatibility both within a separate EMS, and in the future when integrating several EMS in a certain subject area, as well as when building an ILS for training specialists in a certain professional field.

IV. Systematization of educational material in the SET

The main task of the learning process is the formation of a system of knowledge in the student on the subject being studied. In a semantic electronic textbook, this problem is solved through an explicit, visual representation of the semantic structures of the educational material. The formal model of the content structure of the information support of the EMS is presented as a set of sets and structures. For example, a set of semantically elementary information structures of educational and educational material; a set of knowledge base constructions, which are a formalized record of some elementary information construction in the knowledge representation language; set of relationships between the above sets. Such constructions, in particular, include relations that define the description of bibliographic attributes for information constructions, a set of semantic equivalence links between source text fragments and their formalized record, specifications of educational material fragments, and a meta description of educational material. When describing the educational material, one can consider as the main set of objects of study (concepts) of the SEU subject area; a set of statements (such as axioms, theorems, lemmas, etc.) that describe the main properties of the objects of study; a set of relations describing the relationship between the main objects of study. The latter include in particular:

- relations defining the typology of the main objects of research (“subset”, “genus – species”, membership relations, relations “general concept – particular concept”, etc.);
- relations describing the system of concepts of the subject area, based on the hierarchical structure of their definitions (defined concept-defining concepts), etc.

Each concept from the set of objects of the subject area is assigned a knowledge model of the SEU, called the semantic neighborhood of the concept, which is given, in turn, by the formulation from the textbook, as well as the formalized definition of the concept, the relationship between the text of the definition in natural language and the formalized record of the definition); synonyms, homonyms for a given concept; examples - elements of the set denoted by this concept; determining the place of a given concept in the hierarchy of concepts of the theory; a set of relations that are defined on a set of specified concepts; set of the most important statements that describe the properties of this concept. These semantic structures of educational material in the SEU can be used in the following access options:

- Query drafting option: the user generates a query to the system in a specialized query language, or initiates the navigation and search command of the EMS presented in the system menu, in response the system searches for relevant information, for example, searches for the semantic neighborhood of a concept or a section of a textbook;
- a variant of navigation through semantic links, when the student or the system forms a certain path, reveals a feature, moves to the next feature, and so on through the learning material.

V. Knowledge Representation Languages in Semantic Electronic Textbooks and Intelligent Learning Systems

The tools for creating an SET include knowledge representation languages:

1) Basic semantic language SC.
2) Language SCht for representation of hypertext semantic networks. The key nodes of this language are divided into two classes:
   - key nodes that determine the typology of information structures of the hypertext semantic network;
   - key nodes, which are signs of relations, the scope of which includes the specified information constructions.

The SET knowledge representation language is focused on ILS use by both the authors of the educational material and the end users (students), i.e. the language with which the developer structures, systematizes, marks out the educational material and the language of presentation of this material to the student is one and the same language. Accordingly, the means of searching and navigating through the educational material are the same for all categories of users. For navigation and search within the framework of a semantic electronic textbook, special navigation and search tools can be used, which are based on associative access to stored information. The essence of search in graph-dynamic models is to compare the graph-query and fragments of the semantic network. The output to the user of the search results of this or that information about the subject area occurs through the implementation of the corresponding search operations. The execution of each navigation and search operation occurs when the task (request) corresponding to this operation enters the knowledge base. To control the methods for displaying fragments of the hypertext semantic network, it is necessary to introduce the concept of display (reproduction) style and special relationships between the playback style and reproducible fragments. Displaying responses to user requests should be focused on adapting the style of visualization of educational material to the individual characteristics of the student.

The process of designing applied SET should include such stages as the formation of a knowledge base of educational material, test debugging of the system, trial operation.

A semantic electronic textbook, like any other textbook or any book, must contain a description of the structure of the educational material presented, which is reflected in the form of content. Educational material is a structured set of information components of various types. The task of the developer is to isolate these components and describe their order. The presentation of the educational material and, accordingly, the structure of the training course can be linear, or it can also have a branched reading structure.

When writing a textbook, the author immediately focuses on the development of a semantic electronic textbook. A traditional textbook can become part of a semantic electronic textbook (as part of a hypertext semantic network). The construction of a semantic electronic textbook requires the construction of a strict formal presentation of educational material, the systematization and structuring of educational material, a clear consistency of the system of concepts of the subject area, which makes it possible to avoid ambiguity in the understanding of some concepts.

An adaptive approach to designing a user interface in learning systems is one of the promising areas for the development of intelligent learning systems. This approach provides for the creation of a flexible structure of the dialogue between the system and the user in accordance with a number of such individual characteristics of the user as readiness to work with the system, characteristics of interaction with the system, interface preferences, individual psychological characteristics, etc.

VI. The Artificial Intelligence specialty education in the context of creating intelligent training systems in the specialty

The most important direction of improving engineering education, in particular, in the direction of Artificial Intelligence, is the destruction of often artificially created barriers between various academic disciplines, which leads to a "mosaic unsystematic perception of the educational material of the specialty as a whole. Therefore, it is very relevant:

- the transition from programs for individual academic disciplines to comprehensive unified programs for each specialty, where the structuring and systematization of educational material is carried out not on the basis of the conditional division of this material into academic disciplines, but on the basis of ILS semantics;
- transition from textbooks for individual academic disciplines to comprehensive textbooks for each specialty;
- transition from electronic textbooks for individual academic disciplines to complex electronic textbooks and, in general, to intelligent teaching systems for each specialty.

In this regard, it is very important to provide technological means of "transition"of boundaries between educational materials of different academic disciplines. Ideally, the trainee should be able to work with educational material not on the scale of a separate academic discipline, but on the scale of the entire specialty, when solving a number of problems. The principles underlying the consideration of integration issues are based on the general theory of the interaction of scientific and technical disciplines. The SET technology supports the possibility of further mutual integration of the SET in several academic disciplines into a single integrated ILS. Moreover, integration in this case means integration at the content (semantic, semantic) level. The possibility of such integration is primarily provided by the basic language for representing the knowledge of educational material in each individual EMS and in the system as a whole, which makes it possible to describe information at various structural levels, move from level to level and to a meta level, which makes it possible to describe the links between atomic fragments of academic disciplines, between sections of academic disciplines, between the academic disciplines themselves, etc. The integration of many heterogeneous sources of knowledge is carried out on the basis of a single knowledge system, represented as a single conceptual scheme, or ontology. Sources of knowledge can be presented in documentary form (texts), in the form of formatted data (statistical data files), graphic diagrams, expert knowledge (knowledge of specialists). The main requirement for knowledge sources is to prevent the loss and increase the availability of all types of corporate knowledge by providing a centralized, well-structured information repository that meets the requirement of semantic interoperability across disciplines. The structuring of an information warehouse involves the creation and description of a unified knowledge system based on a taxonomy of conceptual concepts, a meta-knowledge base or ontology, through which you can access various sources of knowledge. A number of authors work on this issue within the last decade [6], [7], [8], [9] The integration of SETs in several academic disciplines consists in the integration of hypertext semantic networks of these textbooks and involves:

- coordination of objects of study of these academic disciplines;
- harmonization of the subjects of study of these academic disciplines;
- harmonization of the "foundation"of integrable academic disciplines (basic (undefined) concepts and systems of axioms);
- building interdisciplinary links, which includes the coordination of conceptual systems of integrated academic disciplines; harmonization and integration of the typology of the main objects of research; coordination and integration of the system of statements, about the main objects of research and the relationship between them, etc.;
- integration of relevant scht-constructions;
- pasting of synonymous scht-nodes of the semantic network.

The hypertext semantic network, obtained as a result of the integration of individual SET, will make it possible to localize quite well those groups of concepts that require clarification. With the mutual integration of semantic electronic textbooks, the following problems are solved:

- search for contradictions in the integrated knowledge base;
- maintaining the consistency of constituent elements.

Work on the logical organization of educational material within the specialty, which, in fact, is the preparation of a specialty curriculum, allows you to identify the links between academic disciplines, certain topics of these academic disciplines, their constituent fragments (theorems, definitions of concepts, etc.) with other academic disciplines, topics, fragments of educational material, subsequent and previous. It becomes possible to determine a more rational sequence for studying educational material. As a result, individual SETs in the disciplines of the specialty can be integrated into a complex, an intelligent learning system, which is a hypertext semantic network obtained as a result of the integration of hypertext semantic networks of the SES in all educational disciplines of the specialty. The construction of a semantic electronic textbook requires the construction of a strict formal presentation of educational material, the systematization and structuring of educational material, a clear consistency of the system of concepts of the subject area, which makes it possible to avoid ambiguity in the understanding of some concepts. The analysis of semantic correctness and editing of the knowledge base can be performed by the developer by navigating through the semantic links of the knowledge base of the SET. Explicit description of interdisciplinary links during the integration of SES will allow developing electronic textbooks for a complex of related academic disciplines, including a complex of academic disciplines for the entire specialty. Using the principles underlying the semantic electronic textbook and providing semantic structuring and systematization of stored information will also allow a new approach to solving the problems of intellectualization of computer educational resources and, in particular, educational sites in the open education system. An intellectual learning system (ILS) should be able to track the consistency and integrity of the picture of the world presented in it and presented to the student, teach through ILS own ability to solve problems, contain a system of assessments and decision-making on a learning strategy based on these assessments, i.e. should itself consist of a number of subsystems containing knowledge bases semantically correlated with each other. One of the necessary characteristics of any learning process and, of course, SET is the ability to intelligently assist the student. In educational practice, intellectual assistance means the automation of the teacher’s consulting work, when the student independently solves problems with the support of an automated system. Thus, the IOS helps the user to make decisions by providing relevant information and decision rules in a particular situation. At the same time, in the process of searching for an answer, the user considers various options for solving the problem presented by the knowledge management system, modifies the problem statement or models the situation, choosing the most appropriate solutions. There may be another mode of solving the problem, when the user independently solves the problem, and evaluates the result of the solution with the help of the ILS for correctness and effectiveness based on comparison with the solution proposed by the system itself, or, for example, asking for help directly from the teacher. As specialists involved in filling the knowledge base of the entire ILS, as well as the knowledge bases of its subsystems, in building the subsystems of learning strategies, specialist models, problem solvers, and others, specialists in a particular field of knowledge, pedagogues, methodologists, knowledge engineers should be involved. The trainee, trainer and customer of personnel can act both as users and as conditionally developers of the SET, making adjustments to its work by their actions. The integration of heterogeneous knowledge sources, the interdisciplinary nature of their use, the need to attract additional sources of knowledge, the exchange of knowledge between users involves the development of a knowledge management system architecture based on a common information space in the form of an integrated memory of a virtual university and knowledge ontologies, i.e. based on the properties of interoperability and convergence of systems and knowledge. An educational, educational, training organization or the structure and process of learning is not just a set of automated and intelligent learning systems in certain disciplines that have multimedia tools, flexible learning strategies, subsystems for adapting to the user, etc. For the effective use of all these tools, an infrastructure is needed in which information is processed, interaction between users and subsystems, joint problem solving, in which both users and subsystems are involved. Such a complex system, which combines many autonomous entities (or agents) serving users, solving problems, teaching students, etc., is a multi-agent system (MAS). Any organization that acts as a set of entities that perform certain functions in the interests of achieving the goals of the entire organization is a multi-agent system. All this becomes possible within the OSTIS ecosystem, as an association of structures, objects and their interaction with each other and with the external environment [10].

References

[1] D. Tapscott, *The Digital Economy: Promise and Peril In The Age of Networked Intelligence 1st Edition*. McGraw-Hill: McGraw-Hill, 1st edition, 2014.

[2] D. Bell, *Gryaduschee postidustrialnoye obschestvo: Opyt socialnogo prognozirovaniya[The coming post-industrial society: the experience of social forecasting]*. Mozhaysk, Russia: Moskwa, Akademia [Moscow: Academy], 2004.

[3] A. R.I.Zinurova, "Multimediynye moduli v formate distancionnykh obrazovatelnykh tekhnologij: problema elektronnogo kontenta [Multimedia modules in the format of remote educational technology: The problem of electronic content]," in *Vestnik Kazanskogo Universiteta [Proceedings of the Kazan University]*, vol. 17, no. 12. KGU, Kazan, 2014, pp. 243–246.

[4] A. Baboshkin, "A. baboshkin biznes-modeli obrazovatelnykh proektov [Business models of educational projects]," in *Vestnik molodezhnoy nauki [Bulletin of youth science]*, vol. 11, no. 4. Baltic Federal University I.Kant, Russia, 2017, pp. 5–10.

[5] N. Bezzubenok, "Gibridnye intellektualnye obuchayuschie sistemy na baze gipertkstovykh semanticheskikh setey [Hybrid intellegent educational systems based on hypertext semantic networks]," in *Izvestiya Belorusskoy inzhenernoy akademii [Proceedings of the Belarusian Engineering Academy]*, vol. 15, no. 1. BSUIR, Minsk, 2003, pp. 74–76.

[6] Y. Zhuk, "Otsenka-effektivnosti-raboty-generatora-semanticheskoy-seti-dialogovoy-informatsionnoy-sistemy [Evaluation of the efficiency of the generator of the semantic network of the dialogue information system]," in *Trudy BGTU. Sria 3: Fiziko-matematicheskie nauki i informatika[Proceedings of BSTU. Series 3: Physical and Mathematical Sciences and Informatics]*, vol. 218, no. 1. EE "Belarusian State Technological University", Minsk, Belarus, 2019, pp. 75–78.

[7] N. S.E.Veremchuk, "Sistema-testirovaniya-znaniy-na-estestvennom-yazyke-na-osnove-semanticheskoy-seti-obuchayuschey-sistemy [Natural language knowledge testing system based on the semantic network of the learning system]," in *Trudy BGTU. Sria 3: Fiziko-matematicheskie nauki i informatika[Proceedings of BSTU. Series 3: Physical and Mathematical Sciences and Informatics]*, vol. 218, no. 1. EE "Belarusian State Technological University", Minsk, Belarus, 2019, pp. 51–53.

[8] N. E.V. Smirnova, E.K.Dobritsa, "Ispolzovanie-ontologiy-v-obrazovatelnyh-protsessah [Using ontologies in educational processes]," in *Problemy sovremennoy nauki i obrazovaniya [Problems of modern science and education]*, vol. 104, no. 22. Olymp, 2017, pp. 15–20.

[9] A. E.A. Gavrilina, M.A.Zakharov and E.V.Smirnova, "Ontologicheskiy-podhod-k-testirovaniyu-urovnya-vladeniya-obuchayuschimsya-metapredmetnymi-ponyatiyami [Ontological approach to testing the level of mastery of students in meta-subject concepts]," in *Mashinostroenie i komputernye tekhnologii [Mechanical engineering and computer technology]*, vol. 2, no. 2. MSTU im. Bauman, Russia, 2015, pp. 15–20.

[10] V. Golenkov, N. Gulyakina, I. Davydenko, and D. Shunkevich, "Semanticheskie tekhnologii proektirovaniya intellektual'nyh sistem i semanticheskie associativnye komp'yutery [Semantic technologies of intelligent systems design and semantic associative computers]," *Otkrytye semanticheskie tehnologii proektirovanija intellektual'nyh sistem [Open semantic technologies for intelligent systems]*, pp. 42–50, 2019.

Семантически совместимые OSTIS-системы автоматизации образовательной деятельности

Н.А. Гулякина, Е.И. Козлова

Предлагается класс смысловых электронных учебников, в основе которых лежит смысловое структурирование учебного материала. Благодаря смысловому структурированию учебно-методического материала СЭУ приобретает новые возможности по сравнению с традиционными электронными учебниками. Смысловой электронный учебник представляет собой интерактивный интеллектуальный самоучитель по определенной предметной области, содержащий подробные методические рекомендации по ее изучению и предназначенный для целеустремленного, самостоятельного и активного пользователя, желающего приобрести знания по соответствующей дисциплине (предметной области). На примере специальности «Искусственный интеллект» рассматривается перспектива проектирования интеллектуальных систем обучения для специальностей.

Received 01.11.2022

A semantics-based approach to automatic generation of test questions and automatic verification of user answers in the intelligent tutoring systems

Wenzu Li
Belarussian State University Informatics and Radioelectronics
Minsk, Belarus
lwzzggml@gmail.com

***Abstract*—The article is dedicated to the problem of test question generation and user answer verification in the intelligent tutoring systems. The approach of using knowledge base to automatically generate various types of test questions in the intelligent tutoring systems developed using OSTIS Technology and the approach of realizing automatic verification of user answers based on various semantic structures of described knowledge are introduced in detail in this article.**

***Keywords*—test question generation, user answer verification, OSTIS Technology, intelligent tutoring systems, semantic structure, knowledge base**

I. Introduction

As an activity of the progress and development of human society, education has made a unique contribution to the progress of human civilization, especially with the development of science and technology, education is playing an increasingly important role in modern society. In recent years, with the development of modern information technology such as artificial intelligence, computer researchers have begun to apply artificial intelligence technology to the field of education. The application of artificial intelligence technology in the field of education can not only improve the learning efficiency of learners, but also an important means to ensure the fairness of education. Among them, the most representative product combining artificial intelligence technology and education is the intelligent tutoring systems (ITS) [5].

Compared with the traditional multimedia training system (MTS), ITS has the following characteristics:

- able to conduct free man-machine dialogue;
- providing personalized learning strategies;
- automatic solution of test questions;
- automatic generation of test questions;
- automatic verification of user answers;
- etc.

Among them, automatic generation of test questions and automatic verification of user answers are the most basic and important functions of ITS. It allows automation of the entire process from test question generation, exam paper generation to automatic verification of user answers and scoring of exam papers. This can not only greatly improve the efficiency of testing the user's knowledge level, but also reduce their learning cost, while eliminating human factors to ensure the fairness and justice of the testing process as much as possible.

Although some approaches and systems for automatic generation of test questions and automatic verification of user answers have been proposed and developed by some scientific research teams in recent years with the development of related technologies such as semantic web and natural language processing (NLP), these approaches and systems have many shortcomings, for example:

- only simple objective questions can be generated;
- most of the existing answer verification approaches and systems only support the verification of user answers to objective questions;
- some existing approaches to verifying user answers to subjective questions are based on keyword matching and probability statistics and do not consider the semantic similarity between answers;
- partially semantic-based verification approaches to user answers to subjective questions can only calculate the similarity between answers with simple semantic structures;
- components developed using existing approaches to test question generation and user answer verification can only be used in the corresponding systems and are not compatible with each other;
- automated implementation of the entire process from test question generation to user answer verification is not supported [1], [2], [6].

Objective questions refer to a type of question that has a unique standard answer. In this article, objective questions include: multiple-choice questions, fill in the blank questions and judgment questions. Objective questions differ from subjective questions, which have more than one potential correct answer and sometimes have room for a justified opinion. Subjective questions in this article include: definition explanation questions, proof questions and problem-solving task [8].

For the above reasons, an approach to automatic generation of test questions and automatic verification of user answers in ITS developed using OSTIS Technology (Open Semantic Technology for Intelligent Systems) is proposed in this article, and the implementation process of the approach is described in detail in this article, that is, the development of a universal subsystem for automatic generation of test questions and automatic verification of user answers. The basic principle of automatic generation of test questions in this article is to first summarize a series of test question generation strategies based on the structural characteristics of the ostis-system (system built using OSTIS Technology) knowledge base and the knowledge representation structure therein, and then use these test question generation strategies to extract corresponding semantic fragments from the knowledge base and generate semantic models corresponding to test questions [1], [4]. The basic principle of test question answer verification is to first calculate the similarity between the semantic graph of the standard answer and the semantic graph of the user answer, and then realize the automatic verification of the user answer based on the calculated similarity and the evaluation strategy of the corresponding test question. A semantic graph is a network that represents semantic relationships between concepts. In the ostis-systems, the semantic graph is constructed using SC-code (as a basis for knowledge representation within the OSTIS Technology, a unified coding language for information of any kind based on semantic networks is used, named SC-code) [4], [6]. It should be emphasized that the semantic graph corresponding to the test question and its corresponding natural language description are converted to each other using the natural language interface [7]. The approach proposed in this article needs to solve the following tasks:

- automatic generation of a number of test questions from the knowledge base and storage in the corresponding sections of the subsystem knowledge base;
- design and build subsystem knowledge bases for storing generated test questions;
- according to the needs of users, the corresponding types of test questions are extracted and composed of exam papers;
- calculating the similarity between the semantic graphs of the answers to the objective questions;
- calculating the similarity between the semantic graphs of the answers to the definition explanation questions;
- calculating the similarity between the semantic graphs of the answers to the proof questions and the problem-solving task;
- automatic verification of test question answers and automatic scoring of exam papers based on the calculated similarity and the evaluation strategy of the corresponding test questions.

It should be emphasized here that in the previous articles, we have introduced the implementation process of the corresponding approaches by module (automatic test question generation module and user answer automatic verification module). For example, in the literature [6] we detail the approach to automatically generate various types of test questions from the knowledge base of the ostis-systems, and in the literature [8] we detail the approach to automatically verify user answers in the ostis-systems (including verification of user answers to subjective questions and verification of user answers to objective questions). Therefore, this article focuses on the automation of the entire process from test question generation to user answer verification, and the development of a universal subsystem for automatic generation of test questions and automatic verification of user answer. The approach proposed in this article does not rely on any natural language, but in order to explain how the proposed approach works, the semantic fragments and illustrations selected in this article are presented in English. Among them, the discrete mathematics ostis-system and the euclidean geometry ostis-system will be used as demonstration systems for the subsystem developed using the proposed approach.

II. Existing approaches and problems

A. Automatic generation of test questions

Approach to automatic generation of test questions mainly studies how to use electronic documents, text corpus and knowledge bases to automatically generate test questions quickly and flexibly. Among them, the knowledge base stores highly structured knowledge that has been filtered, and with the development of semantic networks, using the knowledge base to automatically generate test questions has become the most important research direction in the field of automatic generation of test questions [5], [9], [13]. Some of the research results are listed below:

- an approach to using classes, instances, attributes and relationships between them in the OWL ontology for generating multiple-choice questions is presented in the literature [12]. The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent knowledge about things, groups of things, and relations between things. Ontology is a type of knowledge, each of which is a specification of the corresponding subject domain, focused on describing the properties and relations of concepts that are part of the specified subject domain;
- an approach to automatically generate objective questions using an ontology created by Protégé [11] is presented in the literature [10].

These approaches mainly have the following problems:

- the approach of using electronic documents to automatically generate test questions requires a large number of sentence templates;
- the creation of text corpus requires a lot of human resources to collect and process various knowledge;
- existing approaches can only be applied in the corresponding systems and are not compatible;
- existing approaches only allow to generate simple objective questions.

B. Automatic verification of user answers

Automatic verification of user answers is divided into verification of answers to objective questions and verification of answers to subjective questions. The basic principle of verification of answers to objective questions is relatively simple, i.e., it is enough to determine whether the string of the standard answer and the string of the user's answer match. The answers to subjective questions are usually not unique, so the basic principle of verification of answers to subjective questions is to calculate the similarity between standard answers and user answers, and then to implement automatic verification of user answers based on the calculated similarity and the evaluation strategy of the corresponding test questions. The more similar the standard answer and the user answer are, the higher the similarity between them [14], [16], [17]. Verification of answers to subjective questions is divided into the following categories according to the approach used to calculate similarity:

- Based on keyword phrases
  This type of approach first allows to split the sentences into keyword phrases and then calculate the similarity between them according to the matching relationship of keyword phrases between sentences. Representative approaches include:
  - N-gram similarity
  - Jaccard similarity
- Based on vector space model (VSM)
  The basic principle of VSM is to use traditional machine learning algorithms to first convert sentences into vector representations, and then use the distance calculation formula between vectors to calculate the similarity between them [15]. Representative approaches include:
  - TF-IDF
  - Word2vec
  - Doc2Vec
- Based on deep learning
  This type of approach allows the use of neural network models to calculate the similarity between sentences [18]. Representative neural network models include:
  - Tree-LSTM
  - Transformer
  - BERT
- Based on semantic graph
  The basic principle of calculating the similarity between answers (i.e., sentence or short text) using this type of approach is to first convert the answers into a semantic graph representation using natural language processing tools (such as syntactic dependency trees and natural language interfaces), and then calculate the similarity between the semantic graphs (i.e., similarity between answers). In ITS knowledge is stored in the form of semantic graphs, so this type of approach provides the possibility to compute the similarity between any two semantic graphs in ITS. The main advantage of this type of approach is computing the similarity between answers based on semantics. One of the most representative approaches is SPICE (Semantic Propositional Image Caption Evaluation) [19].

These approaches mainly have the following problems:

- the keyword phrase-based approach does not take into account the order between words in a sentence;
- the VSM-based approach leads to the generation of high-dimensional sparse matrices, which increases the complexity of the algorithm;
- semantic graph-based approaches supporting only the description of simple semantic structures;
- these approaches cannot determine whether the sentences are logically equivalent to each other;
- these approaches are dependent on the corresponding natural language.

Therefore based on the existing approaches to automatically generate test questions using knowledge bases, approaches to calculate the similarity between answers using semantic graphs, and OSTIS Technology, an approach to automatically generate test questions and automatically verify user answers using semantics is proposed in this article.

III. Proposed approach

The main task of this article is to detail an approach to automatic generation of test questions and automatic verification of user answers in the ostis-systems and to develop a universal subsystem based on this approach. Where the universality of the subsystem means that the subsystem can be easily transplanted between different ostis-systems. The proposed approach can be divided into two parts according to the functions to be implemented, i.e., automatic generation of test questions and automatic verification of user answers [8]. Therefore, we will introduce the implementation process of these two parts separately.

A. Automatic generation of test questions

The basic principle of automatic generation of various types of test questions (including objective questions and subjective questions) in the ostis-systems is to first extract the corresponding semantic fragments from the knowledge base using a series of test question generation strategies summarized based on the knowledge representation approach and the knowledge description structure in the framework of OSTIS Technology, then add some test question description information to the extracted semantic fragments, and finally store the semantic fragments describing the complete test questions in the corresponding section of the universal subsystem [1]. When exam papers needs to be generated, the subsystem allows to extract some corresponding test questions from the subsystem knowledge base according to the parameters input by the user and combine them into exam papers. The test questions and exam papers in the form of semantic graphs are converted into natural language descriptions using a natural language interface. Since in the literature [6] we have detailed some of the strategies used for automatic generation of test questions in the ostis-systems, we next select only some of the test question generation strategies for introduction.

- Test question generation strategy based on class
  This type of test question generation strategy is used to automatically generate objective questions based on

various relations between classes. It is further divided into:

- Based on "inclusion*" relation
  The inclusion relation is one of the most frequently used relations in the knowledge base of the ostis-systems, which is satisfied between many classes (including subclasses), so that the inclusion relation between classes can be used to generate objective questions. The set theory expression form of inclusion relation between classes is as follows: $S_i \subseteq C(i \geq 1)$, ($S$-subclass, i-subclass number, C-parent class). The following shows a semantic fragment in the knowledge base that satisfies the inclusion relation in SCn-code (one of SC-code external display languages) [1], [4]:

  binary tree
  $\Leftarrow$ *inclusion**:
  directed tree
  $\Rightarrow$ *inclusion**:
  - *binary sorting tree*
  - *brother tree*
  - *decision tree*

  Consider the example of a multiple-choice question generated using this semantic fragment according to the strategy of inclusion relations, which has the natural language form shown below:
  <<Binary tree does not include ( )?>>
  A. directed tree B. brother tree
  C. decision tree D. binary sorting tree
  Similarly, other types of objective questions can be generated using this strategy;
- Based on "subdividing*" relation
  The result of set subdivision is to get pairs of disjoint sets, and the union of these disjoint sets is the original set. The subdividing relation is also an important relation in the knowledge base, so that semantic fragments in the knowledge base that satisfy this relation can be used to generate objective questions;
- Based on "strict inclusion*" relation
  Strict inclusion relation is a special form of inclusion relation ($S_i \subset C(i \geq 1)$). Using strict inclusion relation to automatically generate objective questions is similar to using inclusion relation.

Other strategies used to generate objective questions include:

- Test question generation strategy based on elements;
- Test question generation strategy based on identifiers;
- Test question generation strategy based on axioms;
- Test question generation strategy based on relation attributes;
- Test question generation strategy based on image examples.

The process of generating subjective questions using subjective question generation strategy is as follows:

- searching the knowledge base for semantic fragments describing the definition, proof or solution of the question using logic rules (i.e. templates constructed using SC-code);
- storing the found semantic fragments in the corresponding section of the knowledge base of the subsystem;
- using manual approaches or automatic approaches (such as natural language interfaces) to describe the definition, proof process or solution process of the corresponding test question according to the knowledge representation rules (i.e. standard answers to subjective questions). Among them, standard answers to subjective questions are represented using SCg-code (SCg-code is a graphical version for the external visual representation of SC-code) or SCL-code (a special sub-language of the SC language intended for formalizing logical formulas) [1], [4].

Using these test question generation strategies described above allows various types of test questions to be generated automatically from the knowledge base. These automatically generated test questions are stored in the knowledge base of the subsystem according to their type and the corresponding test question generation strategy, this type of storage allows to quickly and dynamically generate exam papers according to the needs of user needs. In the next section we will describe in detail the construction of the knowledge base of the subsystem and the way in which test questions are stored in it. The proposed approach to generating test questions has the following advantages:

- OSTIS Technology supports uniform knowledge representation approaches and knowledge description structures, so the proposed approach to generating test questions can be used in different ostis-systems;
- the generated test questions are described using SC-code, so they do not rely on any natural language;
- using the proposed test question generation approach, not only objective questions but also subjective questions can be generated.

B. Automatic verification of user answers

In the ostis-systems, test questions are stored in the knowledge base in the form of semantic graphs, so the most critical step of user answer verification is to calculate the similarity between the semantic graph of standard answer and the semantic graph of user answer, and when the similarity is obtained and combined with the evaluation strategy of the corresponding test questions, the correctness and completeness of user answers can be verified [2], [8].

User answer verification is classified according to the type of test questions: 1. verification of answers to objective questions; 2. verification of answers to subjective questions. Although the most critical step of answer verification is all about calculating the similarity between the semantic graphs of answers, the knowledge types (factual knowledge and logical knowledge) and and knowledge structures used to describe different types of test questions are not the same, so the approach to calculate the similarity between the semantic graphs of answers to different types of test questions are different. Factual knowledge refers to knowledge that does not contain variable types, and this type of knowledge expresses facts. Logical knowledge usually contains variables, and there are logical relations between knowledge. In the ostis-systems SCL-code is used to represent logical knowledge. In this article objective questions, proof questions and problem-solving task are described using factual knowledge, and definition interpretation questions are described using factual knowledge and logical knowledge together.

C. Verification of answers to objective question

The semantic graphs used to describe objective types of test questions and their answers in the knowledge base have the same semantic structure, so the similarity between answers to such types of test questions can be calculated using the same approach. Since the user answers in the natural language to the objective questions are already aligned with the existing knowledge in the knowledge base when they are converted to semantic graphs using the natural language interface, that is, the elements representing the same semantics in the knowledge base have the same main identifier (identifier is a file that can be used to denote (name) an entity in the framework of external language) [7]. Therefore, it is not necessary to consider the differences between concepts at the natural language level when calculating the similarity between semantic graphs of answers to objective questions, that is, the similarity between answers is calculated based on the semantic structure. The basic principle for calculating the similarity between semantic graphs of answers to objective questions is shown below:

- the semantic graph of standard answers (s) and the semantic graph of user answers (u) are decomposed into substructures according to the rules of representation of factual knowledge;
- using formulas (1), (2), and (3) to calculate the precision P_{sc}, recall R_{sc} and similarity F_{sc} between semantic graphs.

$$P_{sc}(u,s) = \frac{|T_{sc}(u) \otimes T_{sc}(s)|}{|T_{sc}(u)|} \tag{1}$$

$$R_{sc}(u,s) = \frac{|T_{sc}(u) \otimes T_{sc}(s)|}{|T_{sc}(s)|} \tag{2}$$

$$F_{sc}(u,s) = \frac{2 \cdot P_{sc}(u,s) \cdot R_{sc}(u,s)}{P_{sc}(u,s) + R_{sc}(u,s)} \tag{3}$$

The main calculation parameters in the formulas include:

- $T_{sc}(u)$ — all substructures after the decomposition of the user answers u;
- $T_{sc}(s)$ — all substructures after the decomposition of the standard answers s;
- $\otimes$ — binary matching operator, which represents the number of matching substructures in the set of two substructures.

Once the similarity of the answers is obtained, the correctness and completeness of the user answers to the objective questions can be verified by combining them with the evaluation strategy of the objective questions. The evaluation strategy of the objective questions is shown below:

- if there is only one correct option for the current test question, only if the standard answer and the user answer match exactly, the user answer is considered correct and the user gets the maximum score (Max_{score});
- if the current question has multiple correct options (multiple-choice question with multiple correct options and partially fill in the blank questions):
  - as long as the user answer contains an incorrect option, the user answer is considered incorrect and the user score is 0;
  - if all the options in the user answer are correct, but the number of correct options is less than the number of correct options in the standard answer, the user answer is considered correct but incomplete. At this time, the user answer score is $R_{sc} * Max_{score}$;
  - if all the options in the standard answer match exactly with all the options in the user answer, the user answer is exactly correct, and the user score is Max_{score}.

Fig. 1 shows an example of verification of user answer to subjective question in SCg-code.

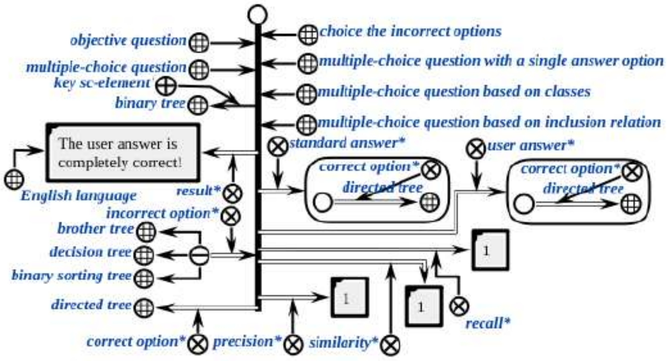


Figure 1. An example of verification of user answer to subjective question.

D. Verification of answers to subjective questions

The most critical step of verification of answers to subjective questions is also the calculation of similarity between semantic graphs of answers, but the knowledge types and knowledge structures used to describe different types of subjective questions and their answers are not the same in the ostis-systems. Thus the approach to calculating the similarity between the semantic graphs of the answers to the subjective questions is further divided into: 1. the approach to calculating the similarity between answers to definition explanation questions; 2. the approach to calculating the similarity between answers to proof questions and problem-solving task.

Calculating the similarity between answers to definition explanation questions

The answers to the definition explanation questions in the ostis-systems are described in the form of logical formulas using factual knowledge and logical knowledge (SCL-code). Logic formulas are powerful tools for formal knowledge representation in the framework of OSTIS Technology, which are expanded based on the first-order predicate logic formulas and inherits all the operational properties of first-order predicate logic formulas [4]. It is to be emphasized that when calculating the similarity between the answers to the definition explanation questions, the factual knowledge in the semantic graph of the user answers has been aligned with the existing knowledge in the knowledge base (using natural language interfaces) [7]. In order to calculate the similarity between the semantic graphs of the answers to the definition explanation questions the following tasks need to be solved:

- automatic selection of potential equivalent standard answer;
- establishing the mapping relationship of potential equivalent variable sc-node pairs between the semantic graphs of the answers;
- calculating the similarity between semantic graphs;
- if the similarity between semantic graphs is not equal to 1, they also need to be converted to the prenex normal form (PNF) representation separately, and then the similarity between them is calculated again [23].

Because some definition explanation questions sometimes have multiple standard answers, but the logical formulas used to represent them formally are not logical equivalents (described according to different conceptual systems). For example, the definition of equivalence relation: 1. in mathematics, an equivalence relation is a binary relation that is reflexive, symmetric and transitive; 2. for any binary relationship, if it is a tolerant relationship and is transitive, then it is an equivalence relation. The logical equivalence between semantic graphs in the ostis-systems is divided into two types: 1. logical equivalence between semantic graphs described based on logical formulas; 2. logical equivalence between semantic graphs based on different conceptual systems. This type of equivalence is further classified according to the type of knowledge:

- logical equivalence between semantic graphs based on factual knowledge;
- logical equivalence between semantic graphs based on logical knowledge (for example, the definition of equivalence relation).

Therefore, when calculating the similarity between answers, it is necessary to filter a standard answer that best matches the user answer from multiple possible standard answers in advance. Therefore an approach to filter a standard answer that best matches the user answer according to the predicate similarity between answers is proposed in this article. This working principle of this approach is shown below:

- finding all the predicates in each answer (non-repeating);
- calculating the predicate similarity between the user answer and each standard answer using formulas (1), (2) and (3);
- the standard answer that is most similar (maximum similarity) to the user answer is selected as the final standard answer.

Since the semantic graphs used to describe the answers to the definition explanation questions are constructed based on logical formulas, the variables sc-nodes (equivalent to the bound variables in the predicate logic formula) are included in the semantic graphs. In order to calculate the similarity between semantic graphs, the most critical step is to establish the mapping relationship of potential equivalent variable sc-node pairs between semantic graphs. Therefore, based on the existing ontology mapping methods and mapping systems (for example, ASMOW, RiMOM, etc.) an approach to establish the mapping relationship of potential equivalent variable sc-node pairs between semantic graphs according to semantic structures (various sc-constructions) are proposed in this article [20], [21], [22].

In the ostis-systems, the sc-construction composed of sc-tuple, relation sc-node, role relation sc-node and sc-connector is used to describe logical connectives (such as negation ($\neg$) and implication ($\rightarrow$), etc.) and quantifiers (universal quantifier ($\forall$) and existential quantifier ($\exists$)), atomic logic formula (various sc-constructions) or multiple atomic logic formulas that satisfy conjunctive relation are contained in the sc-structure and connected with the corresponding sc-tuple, and these sc-elements together constitute the semantic graph used to represent the user answer [1], [22]. All sc-tuples and sc-connectors form a tree, which completely describes the logical sequence between connectives and quantifiers in the logical formula. Because the sc-structure containing the atomic logical formula is connected to the corresponding sc-tuple, as long as the position of each sc-tuple and sc-structure in the semantic graph is determined, the position of each variable sc-node in the semantic graph can be determined. An approach to numbering each sc-tuple and sc-structure in the semantic graph according to a depth-first search strategy (DFS) is proposed in this article. The working process of this approach is shown below:

- starting from the root of the tree structure composed of sc-tuples, each sc-tuple node in the tree is numbered in turn according to the DFS strategy and the priority of the current sc-node (for example, the sc-node priority of the ”if’ ” condition is higher than the sc-node of ”else’ ” conclusion) (the numbering sequence starts from 0);
- according to the numbering sequence of sc-tuple, each sc-tuple in the tree is traversed from small to large, and the sc-structure connected to the current sc-tuple is numbered while traversing (the numbering sequence starts from 1).

For a detailed procedure for numbering sc-tuples and sc-structures, please refer to the literature [8]. In answer verification, if the standard answer and the user answer are exactly equal, it means that the atomic logic formulas with the same semantics between the answers have the same position in the semantic graph (That is, the numbering sequence of sc-structure is the same). Therefore, in this article, the mapping relationship of potential equivalent variable sc-node pairs will be established based on the matching relationship of the sc-constructions in the same position between the answers. The establishment of mapping relationship of the potential equivalent variable sc-node pairs between answers mainly includes the following steps:

1) according to the numbering sequence of the sc-structure in the semantic graph, each time a sc-structure pair with the same number is found from the standard answer and the user answer;
2) according to the priority order (from high to low) of the various types of sc-constructions used to describe the atomic logic formula, it is determined in turn whether the current sc-structure pair contains this type of sc-construction at the same time. If the current sc-structure pair contains this type of sc-construction at the same time, then, according to the matching relationship of each sc-element between the current sc-construction in the standard answer and the current sc-construction in the user answer, the mapping relationship of the potential equivalent variable sc-node pairs between the current sc-construction pair is established;
3) repeat step 1 — step 2 until all mapping relationships between semantic graphs are established [8].

Fig. 2 shows an example of establishing the mapping relationship between semantic graphs in SCg-code.

In Fig. 2, the definition of the inclusion relation is described $(\forall A \forall B((A \subseteq B) \iff (\forall a(a \in A \rightarrow a \in B)))$.

When the mapping relationship between the potential equivalent variable sc-node pairs between the semantic graphs is established, the similarity between the answers can be calculated. The process of calculating the similarity between the semantic graphs of the answers to the definition explanation questions is shown below:

- decomposing the semantic graph of standard answer and the semantic graph of user answer into substructures according to the rules of representation of factual knowledge and logical knowledge;
- numbering the sc-tuples and sc-structures in the semantic graphs of the answers, respectively, and establishing the mapping relationship of potential equivalent variable sc-node pairs between the semantic graphs;
- using formulas (1), (2) and (3) to calculate the precision P_{sc}, recall R_{sc} and similarity F_{sc} between semantic graphs.

Since the semantic graphs of answers to definition explanation questions are described based on logical formulas, if the similarity between semantic graphs is not equal 1 ($F_{sc} < 1$), it is also necessary to determine whether their logical formulas are logically equivalent. There is such a theorem in predicate logic: any predicate logic formula has a PNF that is equivalent to it. Because the logical formulas in the framework of OSTIS Technology are extended based on predicate logical formulas, it also has such a property. Therefore we can consider converting semantic graphs based on logical formula descriptions to PNF descriptions, and then determine whether logical equivalence is satisfied between them [23], [24]. However, the PNF of the logic formula is not unique, and the reasons why the PNF is not unique include:

- the used order of different logical equivalence formulas (conversion rules). For example, converting $(\forall x F(x) \wedge \neg \exists x G(x))$ to PNF:
  - $\forall x F(x) \wedge \neg \exists x G(x)$
    $\Leftrightarrow \forall x F(x) \wedge \forall x \neg G(x)$
    $\Leftrightarrow \forall x(F(x) \wedge \neg G(x))$, (equivalence rule)
  - $\forall x F(x) \wedge \neg \exists x G(x)$
    $\Leftrightarrow \forall x F(x) \wedge \forall y \neg G(y)$, (renaming rule)
    $\Leftrightarrow \forall x \forall y(F(x) \wedge \neg G(y))$, (rule of expansion of quantifier scope)
- the order of the quantifiers in PNF;
- etc.

Therefore, based on the approach to convert predicate logic formulas into PNF and some characteristics of logic formulas in ostis-systems, an approach to convert logic formulas into unique (deterministic) PNF according to strict restriction rules is proposed in this article. The strict restrictions mainly include the following:

- in order to solve the problem that PNF are not unique due to the order in which the logical equivalence formulas are used, we specify that the renaming rule is preferred when converting logical formulas to PNF;
- in order to solve the problem that the PNF is not unique due to the order of the quantifiers, an approach to move all quantifiers to the forefront of the logical formula strictly according to the priority of the quantifiers is proposed in this article. The movement process of quantifiers is shown below:
  - if no quantifiers exist at the front of the logical formula, all existential quantifiers are moved to the front of the logical formula in preference;
  - if the last quantifier at the forefront of the logical formula is a universal quantifier, the universal quantifiers in the logical formula will be moved preferentially to the front of the formula;
  - if the last quantifier at the forefront of the logical formula is a existential quantifier, the existential quantifiers in the logical formula will be moved preferentially to the front of the formula.
- the logical formula used to represent the answer to the definition explanation question can usually be expressed in the following form: $(Q_1 x_1 Q_2 x_2 ... Q_n x_n (A \leftrightarrow B))$, where $Q_i (i = 1, ...n)$ is a quantifier [8], [25]. A is used to describe the definition of a concept at a holistic level, and it does not contain any quantifiers. B is used to explain the semantic connotation of a definition at the detail level, and it is usually a logical formula containing quantifiers (also known as a logical sub-formula). Therefore, based on the characteristics of the logical formula and in order to simplify the knowledge processing, it is only necessary to convert the logical formula B to PNF;

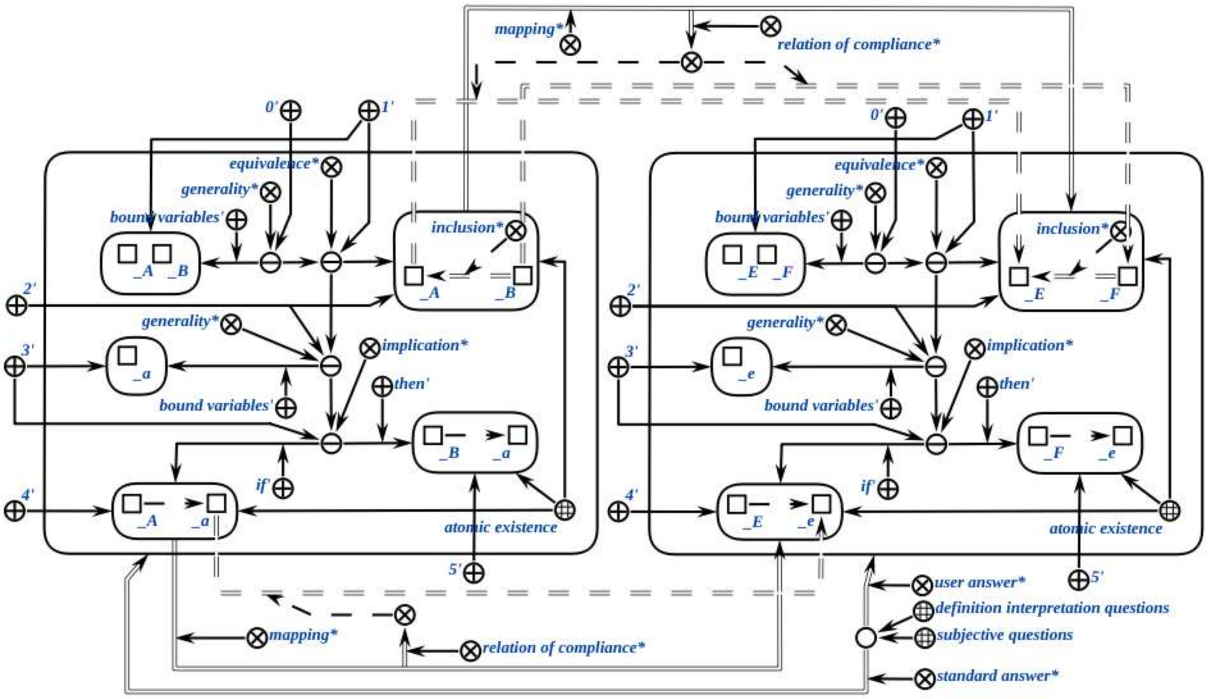


Figure 2. An example of establishing the mapping relationship between semantic graphs.

- to simplify the knowledge processing, only the implication connective need to be eliminated when converting logic formulas to PNF;
- multiple atomic logic formulas connected using the same conjunctive connective are preferentially merged into one whole (i.e. they are merged into the same sc-structure).

The process of converting the semantic graphs of answers to definition explanation questions into PNF descriptions according to strict restriction rules is shown below:

- if there are multiple sc-structures in the semantic graph connected by the same conjunctive connective, the sc-constructions contained in them are merged into the same sc-structure;
- eliminating all the implication connectives in the semantic graphs;
- moving all negative connectives in the semantic graphs to the front of the corresponding sc-structure;
- using renaming rules so that all bound variables in the semantic graphs are not the same;
- moving all quantifiers to the front of the logical formula;
- merging again the sc-structures in the semantic graphs that can be merged.

Fig. 3 shows an example of converting a semantic graph into PNF representation in SCg-code ($\forall A \forall B((A \subseteq B) \leftrightarrow \forall a(a \in A \rightarrow a \in B)) \Leftrightarrow \forall A \forall B((A \subseteq B) \leftrightarrow \forall a(\neg(a \in A) \vee (a \in B))))$.

It should be emphasized that if the calculated similarity between the semantic graphs of PNF representation is not 1 ($F_{sc} < 1$), the similarity between the semantic graphs calculated for the first time is used as the final answer similarity. When the similarity between the answers is obtained and then combined with the evaluation strategy of the subjective questions, the correctness and completeness of the user answers can be verified [8].

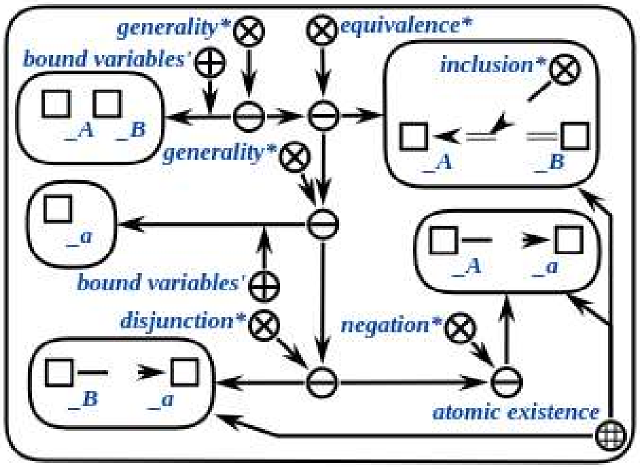


Figure 3. An example of converting a semantic graph into PNF representation.

Calculating the similarity between answers to proof questions and problem-solving task

Both proof questions and problem-solving task in mathematics follow a common task-solving process:

1) the set (Ω) of conditions consisting of some known conditions;
2) deriving an intermediate conclusion using some of the known conditions in Ω and adding it to Ω. Each element in Ω can be regarded as a solving step;
3) repeat step 2) until the final result is obtained [26], [27].

This task-solving process is abstracted as a directed graph, whose structure is in most cases an inverted tree (in special cases the directed graph will contain cycle), and is called a reasoning tree (i. e. the reasoning tree of the standard answer) [26]. Fig. 4 shows an example of a reasoning tree.

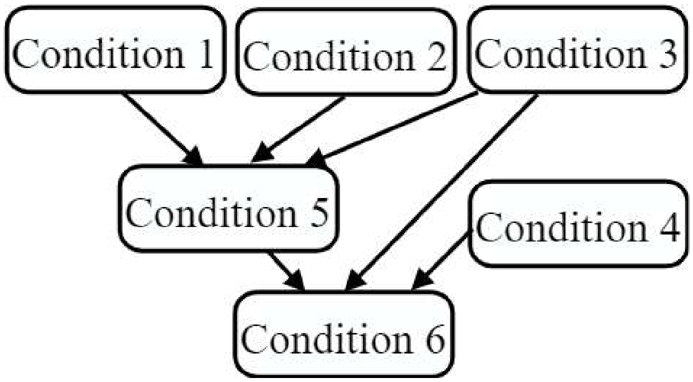


Figure 4. An example of a reasoning tree.

The user answer to the proof question or problem-solving task is a linear structure consisting of some solving steps (i.e. known conditions, intermediate conditions or conclusions), each of which satisfies a strict derivation relationship and logical relationship if the user answer is completely correct. The automatic verification process of user answers to this type of test questions is the same as the traditional manual answer verification process, i.e., verifying whether the current solving step of the user answer is a valid conclusion of the partial solving step preceding that step. This means whether the solving step in the user answer corresponding to the parent node in the reasoning tree always is located after the solving steps in the user answer corresponding to the child nodes.

The semantic graphs of user answers to proof questions and problem-solving task in the ostis-systems are linear structures consisting of some semantic sub-graphs for describing the solving steps and some semantic fragments for describing the logical order and transformation processes between the semantic sub-graphs [1], [4]. The construction process and semantic specification of semantic graphs of user answers to proof questions and problem-solving tasks are described in detail in the literature [3]. The semantic graph of standard answers to this type of test questions is an reasoning tree consisting of a number of search templates (which can be abstracted as the nodes in the tree). Each search template is constructed strictly according to the solving steps of the corresponding test question (i.e., according to the known conditions, intermediate conditions and conclusions in Ω). The search template in the ostis-systems is used to search in the knowledge base for all semantic fragments corresponding to it, and it is constructed based on the SCL-code. The following takes a real problem-solving task as an example to introduce the constructing of the semantic graph of its standard answer (reasoning tree). Description of the problem-solving task: <<Two equal circles externally tangent to other and a third circle the radius of which is 4. The segment that connects the tangent points of the two equal circles to the third circle is 6. Find the radii of equal circles>>. Fig. 5 shows the explanatory picture of the task.

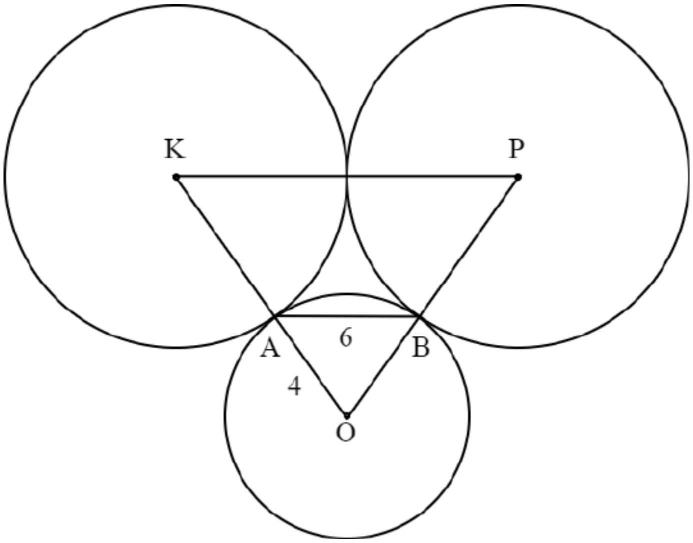


Figure 5. Explanatory pictures for problem-solving task.

Description of user answer in natural language:
1) $\because KP = 2 * R$
2) $\because KO = 4 + R$
3) $\therefore \Delta AOB \backsim \Delta KOP$
4) $\therefore KA = R = 12$

Fig. 6 shows an example of the semantic graph of the standard answer in SCg-code.

The user answers in natural language are converted into semantic graphs using natural language interfaces. Therefore, when calculating the similarity between the semantic graphs of the answers, it is not necessary to consider the differences of the concepts at the natural language level [7]. Fig. 7 shows an example of the semantic specification of a segment in the knowledge base in SCg-code.

From the above example, it can be seen that Segment AB and Segment BA are represented by the same sc-node, they are just two identifiers of the sc-node.

Therefore based on the previously introduced principles of automatic verification of user answers to proof questions and problem-solving task and the semantic models of answers in the ostis-systems, an approach to calculate the similarity between the semantic graphs of answers to proof questions and problem-solving task according to the reasoning tree of standard answer (semantic graph of standard answer) is proposed in this article. The calculation process of similarity between semantic graphs is shown below:

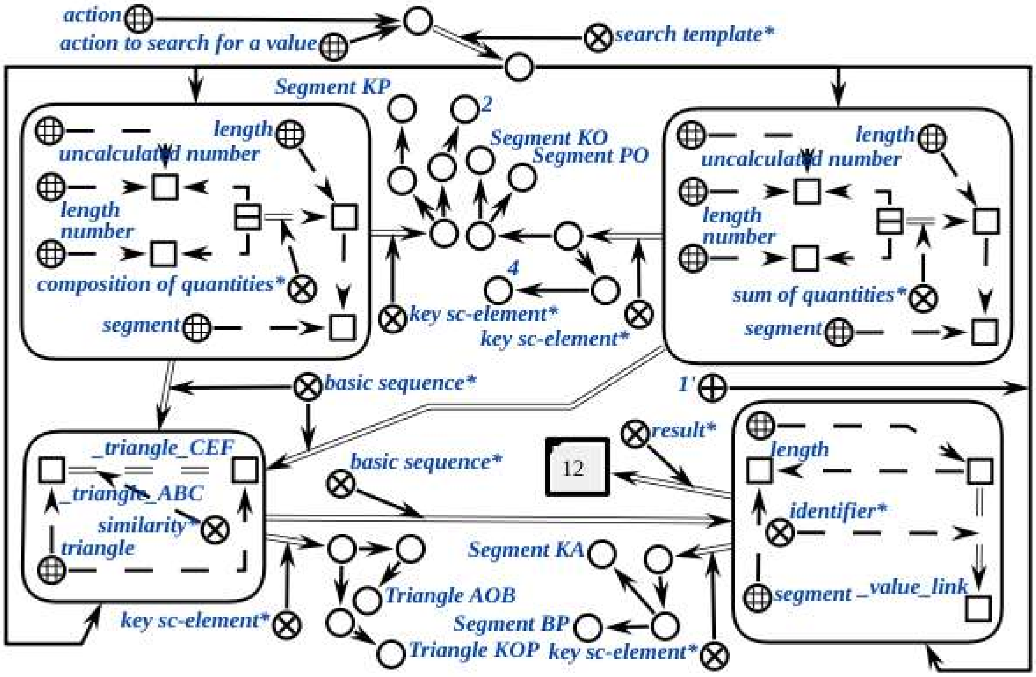


Figure 6. An example of the semantic graph of the standard answer.

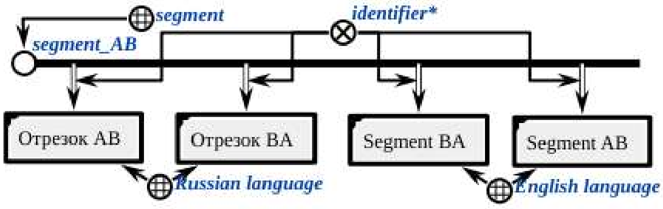


Figure 7. An example of the semantic specification of a segment.

1) numbering each semantic sub-graph (solving step) in the semantic graph of user answers (the numbering order started from 1);
2) each node in the reasoning tree (the search template) is traversed in turn according to the DFS strategy. At the same time, the corresponding semantic sub-graph that is included in the semantic graph of the user answer are searched in the knowledge base using the search template currently being traversed. If such a semantic sub-graph exists, then determine whether the searched semantic sub-graph number is smaller than the semantic sub-graph number corresponding to the search template of the current search template parent node (except for the root node of the reasoning tree), and if so, the searched semantic sub-graph is considered correct;
3) repeat step 2) until all search templates in the reasoning tree have been traversed and the number of correct semantic sub-graphs is counted at the same time;
4) using formulas (1), (2) and (3) to calculate the precision P_{sc}, recall R_{sc} and similarity F_{sc} between answers. Parameters in the formula are redefined:
   - $|T_{sc}(u)|$ — the number of all semantic sub-graphs in the semantic graph of the user answer u;
   - $|T_{sc}(s)|$ — the number of all search templates in the reasoning tree s;
   - $|T_{sc}(u) \otimes T_{sc}(s)|$ — the number of correct semantic sub-graphs.

Once the similarity between the answers to the proof questions and the problem-solving task is obtained, the correctness and completeness of the user answers can be verified combined with the evaluation strategy for the subjective questions.

The evaluation strategy for the subjective questions is shown below:

- if the similarity between the answers is equal to 1 ($F_{sc} = 1$), the user's answer is completely correct and the user gets the maximum score (Max_{score});
- if the similarity between the answers is less than 1 ($F_{sc} < 1$) and the precision is equal to 1 ($P_{sc} = 1$), the user answer is correct but incomplete and the user score is $R_{sc} * Max_{score}$;
- if the similarity between the answers is greater than 0 and less than 1, and the precision is less than 1 ($0 < F_{sc} < 1$ and $P_{sc} < 1$), then the user answer is partially correct and the user score is $F_{sc} * Max_{score}$;
- if the similarity between the answers is equal to 0 ($F_{sc} = 0$), the user answer is wrong and the user score is 0 [8].

The proposed approach to automatic verification of user answers has the following advantages:

- verifying the correctness and completeness of user answers based on semantics;
- the correctness and completeness of user answers to any type of test question can be verified and logical equivalence between answers can be determined;
- allowing the calculation of the similarity between any two semantic graphs in the knowledge base;
- the proposed approach can be used in different ostis-systems.

IV. Knowledge base of the subsystem

The knowledge base of subsystem is mainly used to store automatically generated test questions of various types, and it also allows to automatically extract a series of test questions and form exam papers according to user requirements. Therefore, in order to improve the efficiency of accessing the knowledge base of the subsystem and the efficiency of extracting the test questions, an approach to construct the knowledge base of the subsystem according to the type of test questions and the generation strategy of the test questions is proposed in this article.

The basis of the knowledge base of any ostis-system (more precisely, the sc-model of the knowledge base) is a hierarchical system of subject domains and their corresponding ontologies [1], [3], [4]. Let's consider the hierarchy of the knowledge base of subsystem in SCn-code:

Section. Subject domain of test questions
⇐ *section decomposition**:
{
• *Section. Subject domain of subjective questions*
⇐ *section decomposition*:*
{
• *Section. Subject domain of definition explanation question*
• *Section. Subject domain of proof question*
• *Section. Subject domain of problem-solving task*
}
• *Section. Subject domain of objective questions*
⇐ *section decomposition*:*
{
• *Section. Subject domain of multiple-choice question*
• *Section. Subject domain of fill in the blank question*
• *Section. Subject domain of judgment question*
}
}

Next, taking the subject domain of the objective questions as an example, let us consider its structural specification in SCn-code:

Subject domain of objective questions
∈ *subject domain*
∋ *maximum class of explored objects'*:
objective question
∋ *not maximum class of explored objects'*:
• *multiple-choice question*
• *fill in the blank question*
• *judgment question*

In this article objective types of test questions are decomposed into more specific types according to their characteristics and corresponding test question generation strategies. Next, taking the multiple-choice question as an example let us consider its semantic specification in SCn-code:

multiple-choice question
∈ *maximum class of explored objects'*:
Subject domain of multiple-choice question
⇐ *subdividing**:
{
• *multiple-choice question based on relation attributes*
• *multiple-choice question based on axioms*
• *multiple-choice question based on image examples*
• *multiple-choice question based on identifiers*
• *multiple-choice question based on elements*
⇐ *subdividing*:*
{
• *multiple-choice question based on role relation*
• *multiple-choice questions based on binary relation*
}
• *multiple-choice question based on classes*
⇐ *subdividing*:*
{
• *multiple-choice question based on subdividing relation*
• *multiple-choice question based on inclusion relation*
• *multiple-choice question based on strict inclusion relation*
}
}
⇐ *subdividing**:
{
• *multiple-choice question with multiple answer options*
• *multiple-choice question with a single answer option*
}
⇐ *subdividing**:
{
• *choice the incorrect options*
• *choice the correct options*
}

V. Problem solver

The problem solver of any ostis-system (more precisely, the sc-model of the ostis-system problem solver) is a hierarchical system of knowledge processing agents in semantic memory (sc-agents) that interact only by specifying the actions they perform in the specified memory [1].

Therefore, in order to implement the corresponding tasks, the problem solver for the automatic generation of test questions and automatic verification of user answers is developed in this article, and its hierarchy is shown as follows in SCn-code:

Problem solver for the automatic generation of test questions and automatic verification of user answers
$\Leftarrow$ *decomposition of an abstract sc-agent**:
{
- *Sc-agent for automatic generation of test questions*
  $\Leftarrow$ *decomposition of an abstract sc-agent*:*
  {
  - *Sc-agent for quick generation of test questions and exam papers*
  - *Sc-agent for generating single type of test questions*
  - *Sc-agent for generating a single exam paper*

  }
- *Sc-agent for automatic verification of user answers*
  $\Leftarrow$ *decomposition of an abstract sc-agent*:*
  {
  - *Sc-agent for automatic scoring of exam papers*
  - *Sc-agent for calculating similarity between answers to objective questions*
  - *Sc-agent for calculating the similarity between answers to definition explanation questions*
  - *Sc-agent for converting a logical formula into PNF*
  - *Sc-agent for calculating the similarity between the answers to proof questions and problem-solving task*

  }

}

The main function of the sc-agent for quick generation of test questions and exam papers is to automate the whole process from test question generation to exam paper generation by initiating the corresponding sc-agents (sc-agent for generating single type of test questions and sc-agent for generating a single exam paper). The main function of the sc-agent for generating single type of test questions is to automatically generate a series of test questions from the knowledge base using logical rules built on the basis of SC-code [4]. The logical rules for generating test questions are constructed strictly in accordance with the strategies for generating test questions described earlier. Fig. 8 shows an example of a logic rule for generating multiple-choice question constructed based on a strategy of inclusion relation.

The main function of the sc-agent for automatic scoring of exam papers is to implement automatic verification of user answers to various types of test questions and automatic scoring of exam papers by initiating sc-agents for calculating the similarity between user answers and sc-agents for converting a logical formula into PNF.

VI. Conclusion and further work

A semantic-based approach to automatic generation of test questions and automatic verification of user answers in the ostis-systems is proposed in this article. And based on the proposed approach a universal subsystem for automatic generation of test questions and automatic verification of user answers is developed. The developed subsystem supports automating the entire process from test question generation, to the scoring of exam papers.

The basic principle of automatic generation of test questions is the automatic generation of objective and subjective questions from the knowledge base using some rules constructed based on the structural features of the knowledge base of the ostis-systems. The basic principle of automatic verification of user answers is to first calculate the similarity between the semantic graphs of answers, and then combine it with the evaluation strategy of the corresponding test question to achieve automatic verification of user answers (including the logical equivalence judgment between answers). The proposed approach to calculate the similarity between answers also supports the calculation of the similarity between any two semantic graphs in the knowledge base, so the approach can be used in other tasks in the future as well (such as ontology mapping, knowledge fusion, etc.).

The effectiveness of the developed subsystem will be evaluated in future work.

Acknowledgment

The work in this article was done with the support of research teams of the Department of Intelligent Information Technologies of Belarusian State University of Informatics and Radioelectronics. Authors would like to thank every researcher in the Department of Intelligent Information Technologies.

References

[1] V. V. Golenkov and N. A. Guljakina, “Proekt otkrytoj semanticheskoj tehnologii komponentnogo proektirovanija intellektual’nyh sistem. chast’ 1: Principy sozdanija project of open semantic technology for component design of intelligent systems. part 1: Creation principles],” Ontologija proektirovanija [Ontology of design], no. 1, pp. 42–64, 2014.

[2] V. Golenkov, N. Guliakina, I. Davydenko, and A. Eremeev, ”Methods and tools for ensuring compatibility of computer systems,” in Otkrytye semanticheskie tehnologii proektirovanija intellektual’nyh sistem [Open semantic technologies for intelligent systems], V. Golenkov, Ed., BSUIR. Minsk, BSUIR, 2019, pp. 25–52.

[3] D. Shunkevich, “Metodika komponentnogo proektirovaniya sistem, upravlyaemyh znaniyami,” in Otkrytye semanticheskie tehnologii proektirovanija intellektual’nyh sistem [Open semantic technologies for intelligent systems], V. Golenkov, Ed., BSUIR. Minsk, BSUIR, 2015, pp. 93–110.

[4] (2022, NOV) Ims.ostis metasystem. [Online]. Available: https://ims.ostis.net

[5] Xu G. P., Zeng W. H., Huang C. L. Research on intelligent tutoring system. Application research of computers, 2009, Vol. 26(11), pp. 4020-4030.

[6] Li W., Grakova N., Qian L. Ontological Approach for Question Generation and Knowledge Control. Communications in Computer and Information Science, 2020, Vol. 1282, pp. 161-175.

[7] Qian L., Sadouski M., Li W. Ontological Approach for Chinese Language Interface Design. Communications in Computer and Information Science, 2020, Vol. 1282, pp. 146-160.

[8] Wenzu Li, ”Development of a problem solver for automatic answer verification in the intelligent tutoring systems,” in Otkrytye semanticheskie tehnologii proektirovanija intellektual’nyh sistem [Open semantic technologies for intelligent systems], V. Golenkov, Ed., BSUIR. Minsk, BSUIR, 2021, pp. 169–178.

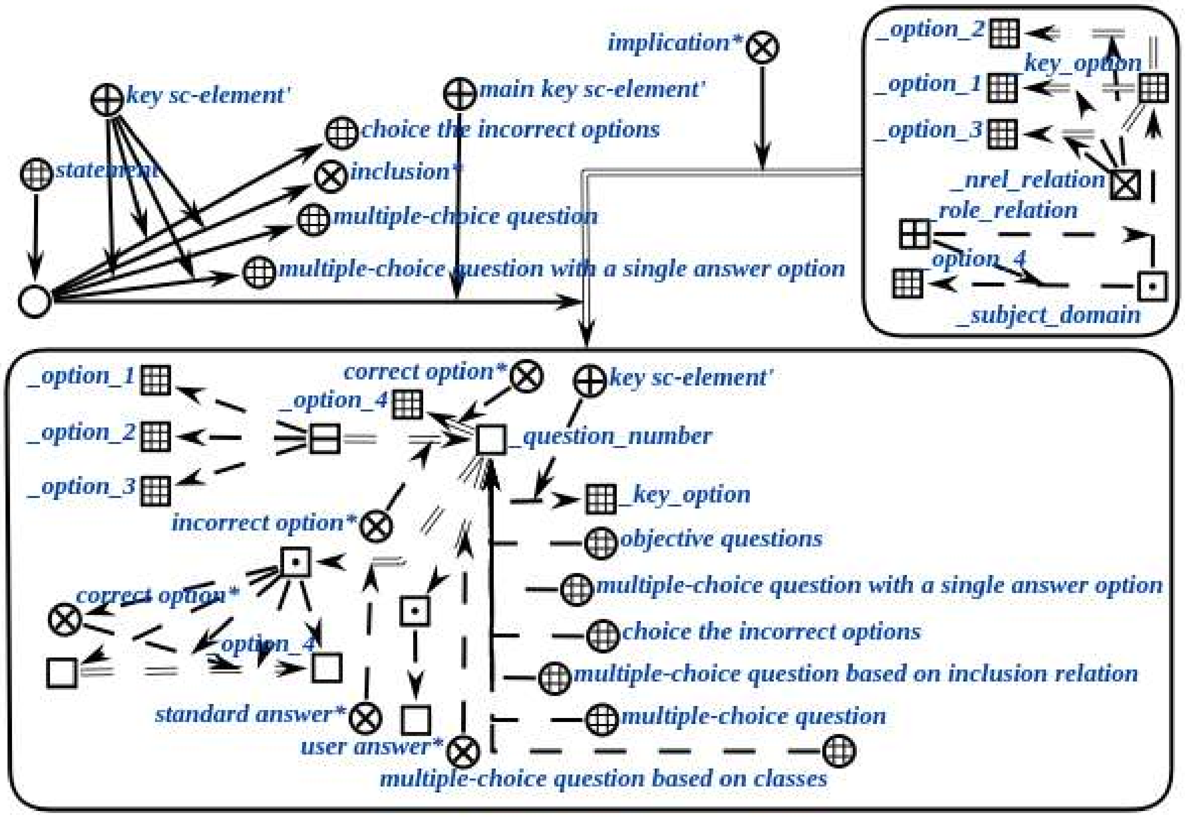


Figure 8. An example of a logic rule for generating multiple-choice question.

[9] Mousavinasab E., Zarifsanaiey N. R., Niakan Kalhori. S. Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods. Interactive Learning Environments, 2021, Vol. 29(1), pp. 142-163.

[10] Li, H. Research on item automatic generation based on DL and domain ontology. Journal of Changchun University of Technology (Natural Science Edition), 2012, Vol. 33(04), pp. 461-464.

[11] (2022, NOV) Protégé [Electronic resource, Online]. Available: http://protege.stanford.edu

[12] Andreas P., Konstantinos K., Konstantinos K. Automatic generation of multiple-choice questions from domain ontologies. In: IADIS International Conference e-Learning, 2008, pp. 427-434.

[13] Arjun S. B., Manas K., Sujan K. S. Automatic Generation of Multiple Choice Questions Using Wikipedia. International Conference on Pattern Recognition and Machine Intelligence, 2013, Vol. 8251, pp. 733-738.

[14] Wan C. L., Yang Y. H., Deng F. A review of text similarity calculation methods. Information science, 2019, Vol. 37(33), pp. 158-168.

[15] Shahmirzadi O., Lugowski A., Younge K. Text similarity in vector space models: a comparative study. 2019 18th IEEE international conference on machine learning and applications (ICMLA), IEEE, 2019, pp. 659-666.

[16] Li X. J. Realization of automatic scoring algorithm for subjective questions based on artificial intelligence. Journal of Jiangnan University (Natural Science Edition), 2009, Vol. 08(03), pp. 292-295.

[17] Wan H. R., Zhang Y. S. Review on Research Progress of Text Similarity Calculation. Journal of Beijing Information Science (Technology University), 2019, Vol. 34(01), pp. 68-74.

[18] Mingyu Ji., Xinhai Zhang. A Short Text Similarity Calculation Method Combining Semantic and Headword Attention Mechanism. Scientific Programming, 2022.

[19] Anderson P., Fernando B., Johnson M., Gould S. Spice: Semantic propositional image caption evaluation. In: European Conference on Computer Vision, Springer, 2016, pp. 382-398.

[20] Su J. L., Wang Y. Z., Jin X. L., et al. Knowledge Graph Entity Alignment with Semantic and Structural Information. Journal of Shanxi University(Nat. Sci. Ed.), 2018, Vol. 42(1), pp. 23-30.

[21] Zhuang Y., Li G. L., Feng J. H. A Survey Entity Alignment of Knowledge Base. Journal of Computer Research and Development, 2016, Vol. 53(1), pp. 165-192.

[22] Wang X. Y., Hu Z. W., Bai R. J., et al. Review on Concepts, Processes, Tools and Methods Ontology Integration. Library and Information Service, 2011, Vol. 55(16), pp. 119-125.

[23] Fujiwara M., Kurahashi T. Prenex normal form theorems in semi-classical arithmetic. The Journal of Symbolic Logic, 2022, pp. 1-31.

[24] Kowalski R. Predicate logic as programming language. In: IFIP congress, 1974, Vol. 74, pp. 544-569.

[25] Pan M., Ding Z. A simple method for solving prenex disjunction (conjunction) normal forms. Computer Engineering and Science, 2008, Vol. 30(10), pp. 82-84.

[26] Jin-zhong Z., Xian-qing H., Xiao-shan G. Automated production of traditional proofs for theorems in euclidean geometry. Annals of Mathematics and Artificial Intelligence, 1995, Vol. 13(1), pp. 109–137.

[27] Zhang J., Peng X., Chen M. Self-evident automated proving based on point geometry from the perspective of Wu's method identity. Journal of Systems Science and Complexity, 2019, Vol. 32(1), pp. 78-94.

ОСНОВАННЫЙ НА СЕМАНТИКЕ ПОДХОД К АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ ТЕСТОВЫХ ВОПРОСОВ И АВТОМАТИЧЕСКОЙ ПРОВЕРКЕ ОТВЕТОВ ПОЛЬЗОВАТЕЛЕЙ В ИНТЕЛЛЕКТУАЛЬНЫХ ОБУЧАЮЩИХ СИСТЕМАХ

Ли Вэньцзу

Данная статья посвящена проблеме генерации тестовых вопросов и проверки ответов пользователей в интеллектуальных обучающих системах. В данной статье подробно представлен подход к автоматической генерации различных типов тестовых вопросов на основе базы знаний в интеллектуальных обучающих системах, разработанных с использованием Технологии OSTIS, и подход к реализации автоматической проверки ответов пользователей на основе различных семантических структур описанных знаний.

Keywords—генерация тестовых вопросов, проверка ответов пользователей, Технология OSTIS, интеллектуальные обучающие системы, онтология, база знаний, семантическая структура

Как деятельность прогресса и развития человеческого общества, образование внесло уникальный вклад в прогресс человеческой цивилизации, особенно с развитием науки и техники, образование играет все более важную роль в современном обществе. В последние годы, с развитием современных информационных технологий, таких как искусственный интеллект, компьютерные исследователи начали работать над применением технологии искусственного интеллекта в сфере образования. Применение технологии искусственного интеллекта в сфере образования может не только повысить эффективность обучения учащихся, но и стать важным средством обеспечения справедливости образования. Среди них наиболее представительным продуктом, объединяющим технологии искусственного интеллекта и образования, являются интеллектуальные обучающие системы (ИОС). Особенно после вспышки COVID-19 в 2020 году была подчеркнута важность и актуальность разработки ИОС. По сравнению с традиционной мультимедийной обучающей системой (МОС), ИОС имеет следующие характеристики:

- способен вести свободный человеко-машинный диалог;
- предоставление персонализированной педагогической услуги;
- автоматическое решение тестовых вопросов;
- автоматическая генерация тестовых вопросов;
- автоматическая проверка ответов пользователей;
- и т.д.

Среди них автоматическая генерация тестовых вопросов и автоматическая проверка ответов пользователей являются самыми основными и важными функциями ИОС. Она позволяет автоматизировать весь процесс от генерации тестовых вопросов, формирования экзаменационных билетов до автоматической проверки ответов пользователей и оценки экзаменационных билетов. Это может не только значительно повысить эффективность тестирования уровня знаний пользователей, но и снизить стоимость их обучения, при этом исключая человеческий фактор, чтобы максимально обеспечить справедливость процесса тестирования.

Received 01.11.22

Ontological approach to batch enterprise within Industry 4.0

Valery Taberko, Dzmitry Ivaniuk
JSC “Savushkin Product”, Brest, Republic of Belarus
Email: id@pda.savushkin.by

***Abstract*—This article provides an review of the current situation of ontology use the techniques of creation, development and practice standards and digital twins with help of OSTIS Technology, examines in more detail the issues of current approaches to standards evolution, maintenance and application, with particular consideration to standards in the field of Industry 4.0, such as ISA-88, ISA-95 and ISA 5.1. Current standards-specific issues in this area are taken into account.**

***Keywords*—Standards, Ontologies, Industry 4.0, OSTIS, ISA-88, ISA-95, ISA-5.1.**

I. Introduction

This work expands on the ideas discussed in [1], [2] and includes descriptions of current issues and new versions of suitable tools for developing and using standards. Also connection with Industry 4.0 is considered - it is typically characterized by its complexity, requiring multidisciplinary knowledge of models, techniques to achieve an integrated solution [3]. With the advent of Industry 4.0, the scenario of reliable and safe interaction of various intelligent systems with each other becomes a reality that technical systems must take into account [4].

Each developed area of human activity is based on a set of standards that formally describe different aspects of it. It includes a system of concepts (including terminology), a typology, and a model, the sequence of actions taken during the process of applying appropriate methods and means. Production site, types and structures of project documents, accompanying activities, etc. The existence of standards allows us to solve one of the key problems associated with any technology. Especially the rapidly developing computer information technology, **compatibility problem** [5] can be solved. Compatibility can be considered in many aspects, from the consistency of terminology in the interactions of process participants to the consistency of actions taken in the process of technology application. On the one hand, the problem with cohesion of digital twin models lies in the fact that a large number of disparate, unrelated and heterogeneous models are required. On the other hand, connecting digital twins in a single system [6] requires their interaction, and awaits conceptual unification of this interaction. It also require from Supervisory Control And Data Acquisition (SCADA) systems a higher level of integration, scalability and technological modernity [7].

Despite advances in information technology, most standards are now presented in the form of traditional linear documents or Web resources containing a series of static pages connected by hyperlinks. This approach to expressing standards has many serious drawbacks, and ultimately the overhead costs of maintaining and using standards actually outweigh the benefits of using them [8].

II. Problems and state of art

An analysis of the work has made it possible to formulate the most important and common problems related to the development and application of modern standards in various fields [8], [9]:

- Above all, the complexity of maintaining the standards themselves due to the duplication of information, especially the complexity of changing terminology.
- Duplicate information in the documentation describing the standard.
- Standards Internationalization Issues – translating a standard into multiple languages actually requires supporting and coordinating independent versions of the standard in different languages.
- As a result, inconsistencies in the format of different standards. As a result, automating the process of developing and applying standards is complicated.
- The inconvenience of using the standard, especially the complexity of finding the information you need. As a result, the complexity of studying standards.
- The complexity of automating the verification that an object or process complies with the requirements of a particular standard.
- etc.

These problems are mainly related to the presentation of standards. The most promising approach to solve these problems is the transformation of each specific standard into a knowledge base, which is based on a set of ontologies corresponding to this standard [5], [8]–[11]. This approach allows us to significantly automate the development processes of the standard and its application.

As an example, consider the ***ISA-88*** [12] standard (the basic standard for batch production). Although this standard is widely used by American and European companies and is actively implemented on the territory of the Republic of Belarus, it has a number of drawbacks listed below. The author's experience with the ISA-88 and ISA-95 standards revealed the following issues related to the versions of the standard:

- The American version of the standard – *ANSI/ISA-88.00.01-2010* – has been updated and is now in its 3rd edition in 2010;
- *ISA-88.00.02-2001* — contains data structures and guidelines for languages;
- *ANSI/ISA-TR88.00.02-2015* – describes an implementation example of ANSI/ISA-88.00.01;
- *ISA-88.00.03-2003* – an activity that describes the use of common site recipes within and across companies;
- *ISA-TR88.0.03-1996* – shows possible recipe procedure presentation formats;
- *ANSI/ISA-88.00.04-2006* – structure for the batch production records;
- *ISA-TR88.95.01-2008* – explains using ISA-88 and ISA-95 together;
- At the same time, the European version approved in 1997 – *IEC 61512-1* – is based on the older version *ISA-88.01-1995*;
- Russian version of the standard – *GOST R IEC 61512-1-2016* – is identical to *IEC 61512-1*, that is, it is also outdated. Also raises a number of questions related to the not very successful translation of the original English terms into Russian.

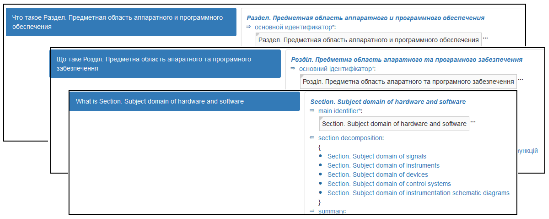
Another standard often used in the context of Industry 4.0 is ***ISA-95*** [13]. ***ISA-95*** is an industry standard for describing high-level control systems. Its main purpose is to simplify the development of such systems, abstract from the hardware implementation and provide a single interface to interact with the ERP and MES layers. Consists of the following parts:

- *ANSI/ISA-95.00.01-2000*, Enterprise-Control System Integration Part 1: "Models and Terminology" – it consists of standard terminology and object models that can be used to determine what information is exchanged;
- *ANSI/ISA-95.00.02-2001*, Enterprise-Control System Integration Part 2: "Object Model Attributes" – it consists of attributes for each object defined in Part 1. Objects and attributes can be used to exchange information between different systems and can also be used as the basis for relational databases;
- *ANSI/ISA-95.00.03-2005*, Enterprise-Control System Integration, Part 3: "Models of Manufacturing Operations Management" – it focuses on Level 3 (Production/MES) functions and activities;
- *ISA-95.00.04* Object Models & Attributes Part 4: "Object models and attributes for Manufacturing Operations Management". The SP95 committee is yet developing this part of ISA-95. This technical specification defines an object model that determines the information exchanged between MES Activities (defined in Part 3 of ISA-95). The model and attributes of Part 4 form the basis for the design and implementation of interface standards, ensuring a flexible flow of cooperation and information exchange between various MES activities;
- *ISA-95.00.05* B2M Transactions Part 5: "Business to manufacturing transactions". Part 5 of ISA-95 is still in development. This technical specification defines operations among workplace and manufacturing automation structures that may be used along with Part 1 and Part 2 item models. Operations join and arrange the manufacturing items and activities described withinside the preceding a part of the standard. Such operations arise in any respect ranges withinside the organisation, however the attention of this technical specification is at the interface among the organisation and the manage system.

Models help define boundaries between business and control systems. They help answer questions about which functions can perform which tasks and what information must be exchanged between applications.

The first phase of building a digital twin model requires embedding data at lower levels of production, such as production processes and equipment. The P&ID production scheme serves as the source of this data. Therefore the ISA 5.1 standard [14] has to work with the P&ID scheme and is widely used in control systems along with the ISA 88 standard to fully describe the lower production levels. This standard is useful when a reference to equipment is required in the chemical, petroleum, power generation, air conditioning, metal refining, and many other industries. The standard enables anyone with a reasonable level of plant knowledge to read flow charts to understand how to measure and control a process without having to go into the details of instrumentation or the knowledge of an instrumentation expert. It is intended to provide sufficient information so that SA5.1 The purpose of this standard is to establish a consistent method of naming instruments and instrumentation systems used for measurement and control. For this purpose, a designation system is presented that includes symbols and identification codes. The latest release from the ISA5.1 subcommittee is the updated *ISA-5.1-2022*, "Instrumentation Symbols and Identification".

Training is an easy way to reach these standards. The International Society of Automation (*ISA*) is a non-profit professional association and recognized leader in automation and control education, dedicated to preparing the workforce for technological change and industry challenges. However, the price is relatively high, around $1,000 per person per day. For 2 persons it is $10,000 for a normal course for 5 days. For some countries it is affordable, for others it is not.



Что такое Раздел. Предметная область аппаратного и программного обеспечения

Раздел. Предметная область аппаратного и программного обеспечения
⇒ основной идентификатор*:
Раздел. Предметная область аппаратного и программного обеспечения

Що таке Розділ. Предметна область апаратного та програмного забезпечення

Розділ. Предметна область апаратного та програмного забезпечення
⇒ основний ідентифікатор*:
Розділ. Предметна область апаратного та програмного забезпечення

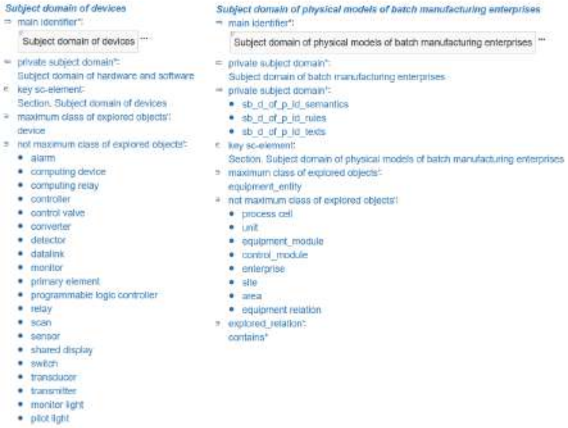
ункцій

What is Section. Subject domain of hardware and software

Section. Subject domain of hardware and software
⇒ main identifier*:
Section. Subject domain of hardware and software
⇐ section decomposition:
{
- Section. Subject domain of signals
- Section. Subject domain of instruments
- Section. Subject domain of devices
- Section. Subject domain of control systems
- Section. Subject domain of instrumentation schematic diagrams

}
⇒ summary:

Figure 1. Start page



Subject domain of devices
⇒ main identifier*:
Subject domain of devices
⇐ private subject domain*:
Subject domain of hardware and software
∈ key sc-element:
Section. Subject domain of devices
⇒ maximum class of explored objects':
device
⇒ not maximum class of explored objects':
- alarm
- computing device
- computing relay
- controller
- control valve
- converter
- detector
- datalink
- monitor
- primary element
- programmable logic controller
- relay
- scan
- sensor
- shared display
- switch
- transducer
- transmitter
- monitor light
- pilot light

Subject domain of physical models of batch manufacturing enterprises
⇒ main identifier*:
Subject domain of physical models of batch manufacturing enterprises
⇐ private subject domain*:
Subject domain of batch manufacturing enterprises
⇒ private subject domain*:
- sb_d_of_p_id_semantics
- sb_d_of_p_id_rules
- sb_d_of_p_id_texts

∈ key sc-element:
Section. Subject domain of physical models of batch manufacturing enterprises
⇒ maximum class of explored objects':
equipment_entity
⇒ not maximum class of explored objects':
- process cell
- unit
- equipment_module
- control_module
- enterprise
- site
- area
- equipment relation

⇒ explored_relation':
contains*

Figure 2. Ontologies for standards (ISA-88, ISA-95 and ISA 5.1)

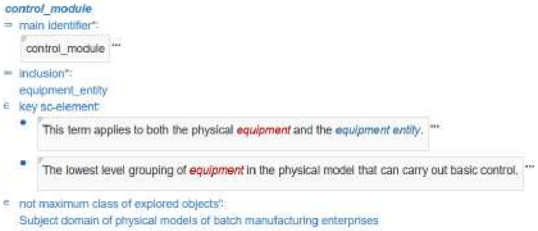
Various established procedural requirements of different organizations are taken into account, but this is done by providing alternative symbology methods unless this conflicts with the goals of the standard. There are many options for adding information or simplifying the symbol if desired.

These and other standards now proliferate in the form of documents that are inconvenient for automated processing and, as noted above, are highly dependent on the language in which each document is written.

III. Examples of system operation with natural language information display

For information to be clear and understandable to the reader, it must be presented in a consistent manner. The recipe authoring system interface allows the structure of domains and ontologies to be expressed in natural language. This process of converting an internal knowledge representation to an external knowledge representation is performed by a graphical interface component. On the main page general information (in 4 languages) is displayed, Fig. 1.

Fig. 2 shows resulting ontologies for standards (ISA-88, ISA-95 and ISA 5.1).



control_module
⇒ main identifier*:
control_module
⇐ inclusion*:
equipment_entity
∈ key sc-element:
- This term applies to both the physical *equipment* and the *equipment entity*.
- The lowest level grouping of *equipment* in the physical model that can carry out basic control.

∈ not maximum class of explored objects':
Subject domain of physical models of batch manufacturing enterprises

Figure 3. Control module

IV. Integration of third-party solutions with a knowledge base

A standard system built on the basis of OSTIS Technology can be easily integrated with other systems in the workplace. To integrate ISA-88, ISA-95 and ISA-5.1 standards system with other systems running on JSC "Savushkin Product", a web-oriented approach is used – the ostis-system server is accessed with the use of the following queries:

```
http :// ostis . savushkin . by?sys_id=
control_module
```

where "sys_id=control_module" defines a term (the name of an entity) whose value we want to find out (in this example, in fact, the answer to the question" What is a "control module"?). This approach makes it relatively easy to add support of the knowledge base for current control systems projects, for this it is enough to indicate the names corresponding to the entities in the knowledge base within the control system. The answer is shown on Fig. 3.

In addition, it is possible to ask more complex and intelligent questions with several arguments, for example, "What is the difference between the concepts of "unit" and "control module"?

The corresponding query to the ostis-system server looks like:

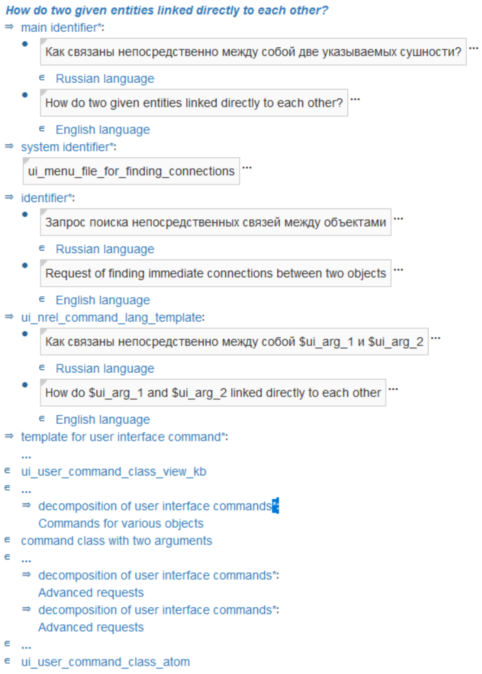
```
http :// ostis . savushkin . by?command_id=
ui_command_difference
&arg1=unit&arg2=control_module
```

Also possible to ask answers questions about questions. Fig. 4 shows result for question "How do two given entities linked directly to each other?"

Thus, an interactive intelligent help system for control systems projects is implemented, allowing employees to simultaneously work with the control system and ask questions to the system directly during the work.

Fig. 5 shows an illustration of the display of information in the form of a HMI page (from the control system project).

Fig. 6 shows a web page that displays the same information as a PFC chart (from the knowledge base).



How do two given entities linked directly to each other?
⇒ main identifier*:
- Как связаны непосредственно между собой две указываемых сущности? ...
  ∊ Russian language
- How do two given entities linked directly to each other? ...
  ∊ English language

⇒ system identifier*:
ui_menu_file_for_finding_connections ...
⇒ identifier*:
- Запрос поиска непосредственных связей между объектами ...
  ∊ Russian language
- Request of finding immediate connections between two objects ...
  ∊ English language

⇒ ui_nrel_command_lang_template:
- Как связаны непосредственно между собой $ui_arg_1 и $ui_arg_2 ...
  ∊ Russian language
- How do $ui_arg_1 and $ui_arg_2 linked directly to each other ...
  ∊ English language

⇒ template for user interface command*:
...
∊ ui_user_command_class_view_kb
∊ ...
  ⇒ decomposition of user interface commands*:
  Commands for various objects
∊ command class with two arguments
∊ ...
  ⇒ decomposition of user interface commands*:
  Advanced requests
  ⇒ decomposition of user interface commands*:
  Advanced requests
∊ ...
∊ ui_user_command_class_atom

Figure 4. System answer on question about question [15].

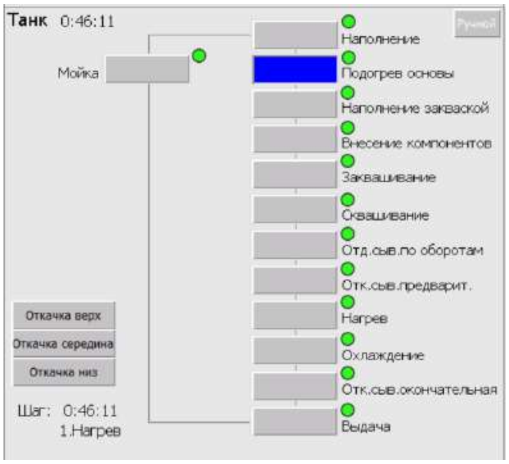


Figure 5. Example HMI from SCADA [15].

Another example is the integrated help subsystem within corporate Add-In EasyEPLANner [16] for CAD EPLAN. It helps to describe technological objects (Tank, Boiler, etc.), operations, etc. according to the ISA-88 standard. Fig. 7 shows a short preview of the project functionality.

Fig. 8 shows UML-model of EasyEPLANner objects to be described in OSTIS.

Fig. 9 shows UML-model of EasyEPLANner control modules to be described in OSTIS.

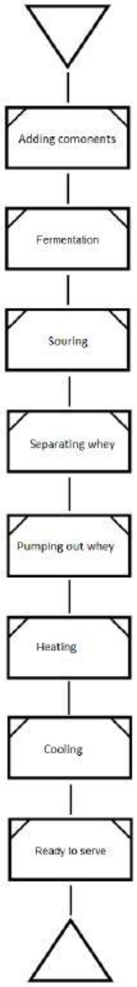


Figure 6. Corresponding PFC chart from OSTIS.

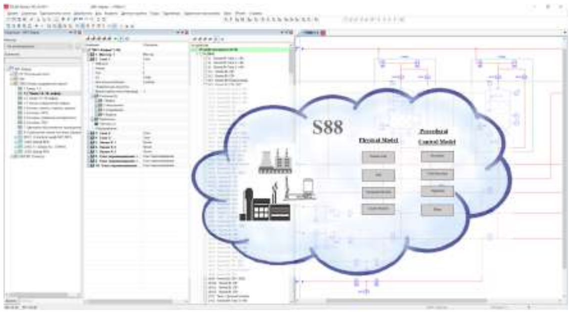


Figure 7. Add-In project EasyEPLANner

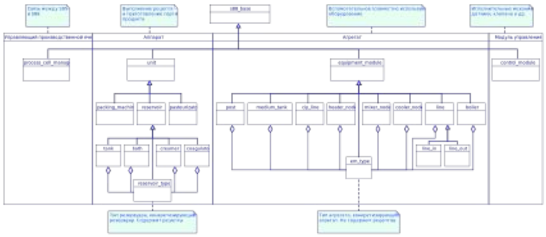


Figure 8. EasyEPLANner objects

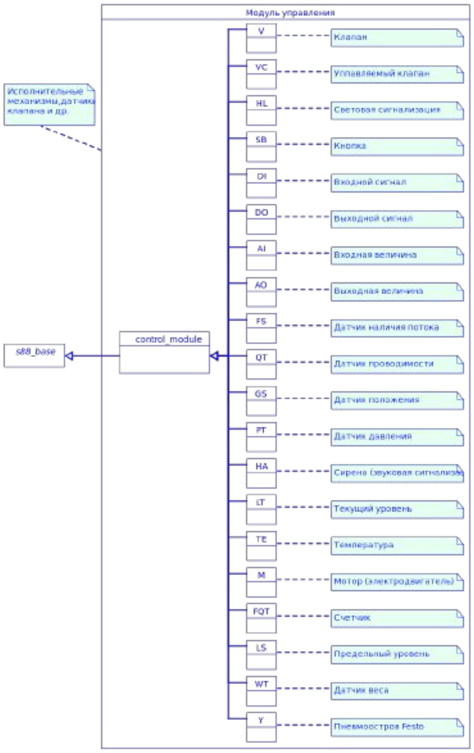


Figure 9. EasyEPLANner objects

V. Use in control system

It is very important to correct and fast react on different events during process control, especially on critical accidents. But when we have complex distributed system it is rather complicated and in normal way require help of the human operator. It may leads to variety of problems. So usage OSTIS-based system can helps to solve as described on Fig. 10. Project #3 has a valve failure but the project does not know what to do. Then it makes a request to the OSTIS server, which already knows which projects also use this line (with this valve). The OSTIS server polls the rest of the projects (projects #1 and #2). Each project has information about which operations are currently active and gives an answer on what to do - pause the operation, do nothing, etc. After that OSTIS-server sends back to project #3 answer with result actions to be used. These are going in automatic way - no need of human operator.

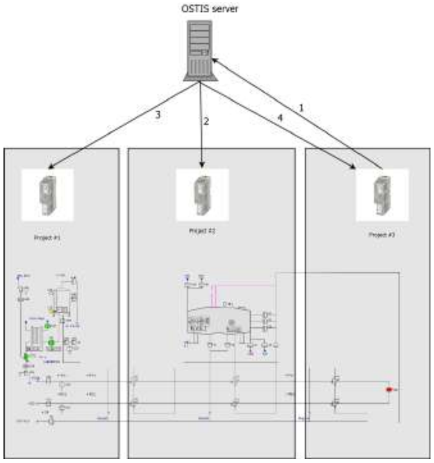


Figure 10. OSTIS in control system

VI. Future development

Current project issues can be found on GitHub ([17], [18] and [19]). Main problems to be solved are:

- Improving system performance and especially accelerating system response time to user requests. It is connect with productivity and overall user satisfaction.
- Continuous updating and refactoring ontological models (further formalization of missing concepts, fix typos and etc.);
- Enhancing PFC-visualisation - not only displaying, but also editing diagrams. Adding rich navigating between PFC-diagram and according text representation;
- Further formulation of questions (typical) to the system from the user and their formalization at the level of the existing knowledge base;
- Adding more description of parts of real control projects based on the existing knowledge base.

The implementation of answers to complex questions is necessary to make easier the work of not only process operators, but also maintenance personnel - instrumentation engineers, mechanics, electricians, etc. Therefore, it is planned to implement the system's answer to the question of the following type - in what operations of which objects this actuator is used (for example, valve "T1V1"). This question is very important when a device failure occurs and it is necessary to determine the criticality of this situation. For analysis, it is necessary to compare the time of the accident and the history of operations. Since, for example, an accident of the mix-proof valve during the line washing operation and the active product dosing along the other line, should lead to a stop of these operations and stop the preparation of the batch in the corresponding unit. The operator must report this to the appropriate maintenance specialist to fix it. After the fault has been eliminated, the operator continues to perform operations. This is the correct events order, which is very important to avoid mixing of detergent and product. If the device malfunction occurred within the line, which is now inactive, then this situation has a low priority, does not lead to a stop in operations and can be eliminated later if the service personnel have free time.

VII. Conclusion

The paper considers an technique to automating the process of creating, developing and making use of standards primarily based on OSTIS Technology. Using the instance of the ISA-88, ISA-95 and ISA-5.1 standards used on the Savushkin Product enterprise, the structure of the knowledge base, the features of the problem solver and the user interface of the support system for these processes are considered. It is proven that the developed system can be easily integrated with other enterprise systems, being the basis for building an information service system for employees in the context of Industry 4.0. The approach proposed in the work allows us to provide not only the ability to automate the processes of creation, agreeing and development of standards, but also allows us to significantly increase the efficiency of the processes of applying the standard, both in manual and automatic way.

Acknowledgment

Authors would like to thank to the research teams of the Departments of intelligent information technologies of the Belarusian State University of Informatics and Radioelectronics and the Brest State Technical University.

References

[1] N. Lutska, O. Pupena, A. Shyshak, V. Taberko, D. Ivaniuk, M. O. Nikita Zotov, and L. Vlasenko, "Ontological model of digital twin in manufacturing," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, ser. 5, V. Golenkov, Ed. BSUIR, Minsk, 2022, p. 310–335.

[2] V. V. Taberko, D. S. Ivaniuk, D. V. Shunkevich, and O. N. Pupena, "Principles for enhancing the development and use of standards within Industry 4.0," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, ser. 4, V. Golenkov, Ed. BSUIR, Minsk, 2020, pp. 167–174.

[3] S. Gil, G. D. Zapata-Madrigal, and G. L. Giraldo-Gómez, "An ontological model to integrate and assist virtualization of automation systems for industry 4.0," *Smart and Sustainable Manufacturing Systems*, vol. 5, p. 10, 09 2021.

[4] V. R. Sampath Kumar, A. Khamis, S. Fiorini, J. L. Carbonera, A. Olivares Alarcos, M. Habib, P. Goncalves, H. Li, and J. I. Olszewska, "Ontologies for industry 4.0," *The Knowledge Engineering Review*, vol. 34, p. e17, 2019.

[5] V. Golenkov, N. Gulyakina, I. Davydenko, and D. Shunkevich, "Semanticheskie tekhnologii proektirovaniya intellektual'nyh sistem i semanticheskie associativnye komp'yutery [Semantic technologies of intelligent systems design and semantic associative computers]," *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, pp. 42–50, 2019.

[6] "Digital Twins for Industrial Applications, an Industrial Internet Consortium White Paper," https://www.iiconsortium.org/pdf/IIC_Digital_Twins_Industrial_Apps_White_Paper_2020-02-18.pdf, 2018.

[7] M. Sverko, T. G. Grbac, and M. Mikuc, "Scada systems with focus on continuous manufacturing and steel industry: A survey on architectures, standards, challenges and industry 5.0," *IEEE Access*, vol. 10, pp. 109 395–109 430, 2022.

[8] P. Serenkov, V. Solomaho, V. Nifagin, and A. Minova, "Koncepcija infrastruktury standartizacii kak bazy znanij na osnove ontologij [the concept of a standardization infrastructure as an ontology-based knowledge base]," *Novosti. Standartizacija i sertifikacija. [News. Standardization and certification.]*, vol. 1, no. 5, pp. 25–29, 2004.

[9] V. Uglev, "Aktualizacija soderzhanija standartov proektirovanija slozhnyh tehnicheskih ob'ektov: ontologicheskij podhod [updating the content of design standards for complex technical objects: ontologic approach]," *Ontologija proektirovanija. [Ontology of designing]*, vol. 1, no. 1, pp. 80–86, 2012.

[10] C. Dombayci, J. Farreres, H. Rodríguez, A. Espuña, and M. Graells, "Improving automation standards via semantic modelling: Application to ISA88," *ISA Transactions*, vol. 67, 01 2017.

[11] M. Vegetti and G. Henning, "Isa-88 formalization. a step towards its integration with the isa-95 standard," in *6th Workshop on Formal Ontologies meet Industry*, vol. 1333, 02 2015.

[12] "ISA-88 standard," https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa88/, (accessed 2022, October).

[13] "ISA-95 standard," https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95/, (accessed 2022, October).

[14] "ISA5.1 Standard," https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa5-1/, (accessed 2022, October).

[15] V. V. Taberko, D. S. Ivanjuk, V. V. Golenkov, I. T. Davydenko, K. V. Ruseckij, D. V. Shunkevich, V. V. Zaharov, V. P. Ivashenko, and D. N. Koronchik, "Ontological design of prescription production enterprises based on ontologies," in *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, ser. 1, V. Golenkov, Ed. BSUIR, Minsk, 2017, pp. 265–280.

[16] "EasyEPLANner project on GitHub," Available at: https://github.com/savushkin-r-d/EasyEPLANner/, (accessed 2022, Jun).

[17] "EasyServer-4.0 project on GitHub," Available at: https://github.com/savushkin-r-d/EasyServer-4.0, (accessed 2022, Jun).

[18] "s88-ostis project on GitHub," Available at: https://github.com/savushkin-r-d/s88-ostis, (accessed 2022, Jun).

[19] "isa-5.1-ostis project on GitHub," Available at: https://github.com/savushkin-r-d/isa-5.1-ostis, (accessed 2022, Jun).

Онтологический подход к рецептурному предприятию в рамках Индустрии 4.0

Таберко В.В., Иванюк Д.С.

В работе рассмотрен онтологический подход к пониманию, интеграции и развитию стандартов на основе Технологии OSTIS. Уточнена формальные трактовки основных понятий, используемых в стандартах, что позволяет упростить описание реальных задач. Также описаны варинты интеграции базы знаний в используемые программные средства разработки и сценарии её использования непосредственно в системах управления.

Received 20.10.2022

Software-Technological Complex for Adaptive Control of a Production Cycle of Robotic Manufacturing

Viktor Smorodin
Department of Mathematical Problems
of Control and Informatics
Francisk Skorina Gomel State University
Gomel, Belarus
Email: smorodin@gsu.by

Vladislav Prokhorenko
Department of Mathematical Problems
of Control and Informatics
Francisk Skorina Gomel State University
Gomel, Belarus
Email: snysc@mail.ru

***Abstract*—An approach is being proposed for constructing a new generation intellectual system based on OSTIS technology for decision making during realization of adaptive control procedures for technological cycle of robotic manufacturing based on the means of software-hardware coupling. At the basis of the decision making intellectual system lays the idea of using neural network controllers that solve the task of searching for optimal maintenance strategy for a technological cycle of robotic manufacturing. A formalization of such a system is being proposed based on OSTIS technology implementation.**

***Keywords*—technological production process, parameters of operation, probabilistic network graph, state indicators, adaptive control, neural network, LSTM, policy-gradient, reinforcement learning**

I. Introduction

The modern convergence direction of research in the sphere of intellectual systems development [1] requires creation of the corresponding software with elements of cognition based on semantically compatible artificial intelligence technologies. Such a direction must also include the creation of computer systems, that provide intellectualization of making analytical decisions, which is directly related to adapting control processes for complex technological systems (technological objects) in real time, creation of semantically compatible knowledge bases in the sphere of analysis of dynamic systems operation and optimization of complex technical systems operation based on them through the creation of open source software for intellectual decision making systems.

When constructing a software solution for solving complex control-related tasks it is important to have a universal interface that would provide semantic compatibility for its elements, allowing their interchangeability and independent development with simple integration. OSTIS technology can serve as a means of achieving this goal and providing a universal platform for connecting various separate problem solvers [2].

Technological systems that can be formalized as probabilistic network graph structures and mathematical models of semi-Markov processes are the object under study in this paper [3].

Adaptive control of a technological cycle is meant as the ability of a control system adequately react on external disturbances and standard control effects with changing the corresponding parameters of control during the system operation.

Optimal control in the scope of this paper is meant as a formalized by a neural network structure of an adaptive control of a technological cycle that is constructed in the base nodes of a probabilistic network graph structure or semi-Markov network model within the chosen quality criteria. The formalization of the control system and mathematical models of the object under study is based upon the authors' scientific research and development in the sphere of simulation modeling of complex technological systems [4].

In this paper a task of optimal maintenance strategy search is being considered for a technological cycle with implementation of reinforcement learning methods based on the selection of criteria chosen by user. An approach is proposed for solving such tasks based on neurocontroller usage that is trained sing policy gradient methods [5].

II. Formalized description of a technological cycle

A technological cycle is understood as a sequence of actions and operations, based on which the manufacturing of products is achieved. There are N technological nodes $(machines)$ in cycle M_i . During the execution of a cycle K operations O_j are being run sequentially. For each operation are given the execution time $t_{(O_i)}$, the set of nodes $\{M_{ijk}\}$, that operate in mode r_j (M_i) for the current operation. The set and content of such operations are defined by the corresponding technological production process.

During the execution of O_j operation an equipment failure of the i-th type node can occur, which demand pausing the cycle and performing maintenance and repair actions. The costs for repairing the i-th type node CM_i and costs for liquidating the consequences of it's failure during the cycle operation CMO_i are given.

When the i-th type node fails during the execution of the corresponding operation it is possible that an emergency may occur. The costs for repairing the i-th type node in case of emergency CE_i and the costs for liquidating the consequences of emergency CEO_i are given.

Before the cycle execution has started maintenance actions may be performed - one or more of the nodes may be checked and repaired.

The maintenance of all kinds is performed by the manufacturing facility personnel, that has a corresponding qualification. The available trained personnel is a limited resource and no more than L nodes can be repaired at the same time. In case a repair is necessary and the required personnel in unavailable, it is necessary to wait until one of the current repair operations ends. The repair times are not static and of probabilistic nature. The costs for cycle not operating (as a result of nodes failure, repair operations, maintenance, liquidation of emergencies) are also given - $CI(T)$ for the non operating period T.

It is assumed that the technological production cycle has integration of means of software-hardware coupling that allow transferring node observation data into the control system when cycle operates, as well as a system of processing recommendations for cycle maintenance that are produced by the control system.

III. Description of the simulation model that is used in this paper

For implementation of a simulation model in the given formalization the data is used:

- distributions for the duration of non-failure operation for the nodes of i-th type $F_{ir}(t_{wf})$ in mode $r(M_i)$;
- distributions for the restoration (repairs) for the nodes of i-th type after a failure $F_{if}(t_r)$;
- distributions for the liquidation of emergency for the nodes of i-th type $F_{ife}(t_{re})$;
- probabilities of emergency during a failure for the nodes of i-th type P_{ie}.

The simulation model operates during the given time period, it restarts the production cycle and, possibly, performs the maintenance actions before each start. The technological cycle control system is being used to make decisions regarding the necessity and contents of the maintenance procedure.

The data describing the current condition of the technological cycle nodes - duration of non-failure operation for all node M_i is passed inside the control system. Based on the control system recommendations the maintenance for the nodes is performed.

IV. Possible approaches to solving the task

When considering a task of such type the method for constructing an optimal strategy is not obvious which makes the usage of traditional supervised learning algorithms problematic. The complex structure and the nature of the possible solutions space in this task make it sensible to consider the reinforcement learning group of algorithms.

Analysis of the modern state of developments in the artificial intelligence field demonstrates that two most effective groups of reinforcement learning algorithms exist for solving complex control tasks:

- value-based - when controller is trained to estimate the future rewards for the actions it selects;
- policy-based - when controller is trained to predict distribution of actions that would lead to the choice of optimal action-selection policy.

In this paper a policy gradient neural network controller will be used for the task under consideration.

Implementation of the reinforcement learning methods implies construction of an environment in which the agent performs actions. The agent selects actions based on the current observations of the environment, and based on the actions performed and the possible changes in the state of the environment a reward is being calculated and may be observed by the agent.

In this paper the environment in which the agent operates is the control system of the technological production cycle that makes available of agent's observation of all nodes M_i non-failure operation duration. Based on the agent's action selection the decision making system forms requests for the maintenance of the technological cycle nodes. When the agent is being trained jointly with the simulation model reward calculation is also done.

V. Shaping the reward function

The reward shaping plays in important role, as it defines the agent's behavior that it learned during training. The choice of the reward function allows to select for optimization the criteria that user prioritizes.

The approach used in this paper includes into reward shaping such components as cycle non-failure operation time (R_{nop}), total sum of maintenance and emergency liquidation costs (R_{cost}), total number of nodes failures (R_f), including the ones that resulted in emergency (R_{fe}), total number of maintenance performed per cycle (R_{rep}). Each of the reward components is present in the equation with a weight coefficient α_i, which characterizes the importance of the component.

During the agent training the value of the reward function is calculated as following:

$$R = \alpha_1 R_{nop} + \alpha_2 R_{cost} + \alpha_3 R_f + \alpha_4 R_{fe} + \alpha_5 R_{rep}$$

VI. Policy gradient

In the policy-based methods instead of the approximation of a numeric function that estimates rewards that agent receives from the environment as a result of his actions, the policy function for action selection is being constructed directly, that connects environment states with agent's actions. The action selection policy is parameterized by the trained parameters of the model that is used to control agent.

Numeric function (reward function) in this case can be used to optimize policy regarding the trained parameters but is not used for action selection. Stochastic action selection policy gives the probability distribution for the possible actions. Such policies are often used in the partially-observable environments when uncertainty exists.

It was shown that for some classes of tasks the policy based methods converge faster than value-based (Q-learning), and also are preferable when the action selection space is of large dimension [5]. The convergence towards at least the local quality maximum is guaranteed.

Policy π is parameterized by the trainable parameters θ.

$\pi_\theta(a|s) = P[a|s]$

This policy returns distribution of actions a when the observable state of the environment is s.

In order to find the values for trainable parameters an optimization problem must be solved for the quality estimation function $J(\theta)$.

$J(\theta) = E_{\pi\theta}(\sum \gamma r)$

Rules to update trainable parameters on the t step:

$\theta_{(t+1)} := \theta_t + \alpha \bigtriangledown J(\theta_t)$

According to the Policy Gradient Theorem[6]

$\bigtriangledown E_{\pi\theta}(r(\tau)) = E_{\pi\theta}(r(\tau) \bigtriangledown log\pi_{\theta(\tau)})$,

which can be transformed as

$\bigtriangledown E_{\pi\theta}(r(\tau)) = E_{\pi\theta}(r(\tau)(\sum_{t=1}^{T} \bigtriangledown log\pi_\theta(a_t|s_t)))$.

The REINFORCE algorithm that in meant to train the agent to perform actions according to the policy that results in maximization of the future rewards could be written like this [5]:

1. Initialize parameters θ

2. Generate an episode in which agent interacts with the environment with $\{S_i\}$, $\{A_i\}$, $\{R_i\}$ – sequences of length T of the observed environment

3. For each step t calculate discounted reward

$G_t := \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

4. Update the parameters by a rule (perform a gradient ascent)

$\theta := \theta + \alpha\gamma^t G \bigtriangledown_\theta ln\pi(A_t|S_t, \theta)$

5. Repeat steps 2-4

VII. Neural network structure choice

For the agent control recurrent neural network based on multi=layer perceptron with LSTM block is being used. As we are working with policy gradient the network output must return the distribution of action probability, thus softmax is used. Network structure:

1) Dense x64 ReLU;
2) Dense x64 ReLU;
3) LSTM x32 ReLU;
4) Dense x6 Softmax.

VIII. Results of the training

One cycle execution in the normal condition takes 48 units of model time. One simulation lasts for 64*48 = 3072 units of model time.

On figure 1-5 the graphs show how various metrics change during the training that lasts 500 episodes.

Distribution of the most frequently produced by the system recommendations for maintenance (7) corresponds with the ones expected based on the chosen simulation parameters for

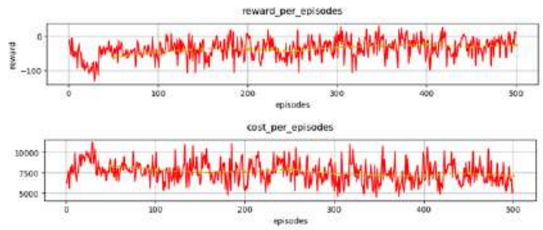


Figure 1. Total reward that agent receives during one simulation run. Total costs for executing the cycle during one simulation during training. A tendency to the increase of the reward and decrease of the costs can be observed during training.

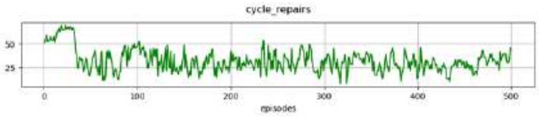


Figure 2. Number of maintenance operations performed according to the system's recommendations.

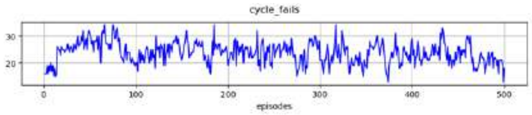


Figure 3. Number of failures during the simulations

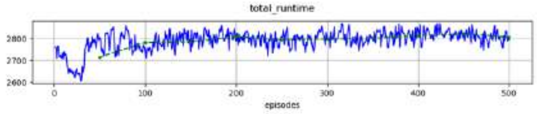


Figure 4. The average time of normal operation of the cycle during simulation.

Figure 5. Distribution of recommendations for maintenance, that are most frequently generated by the system.

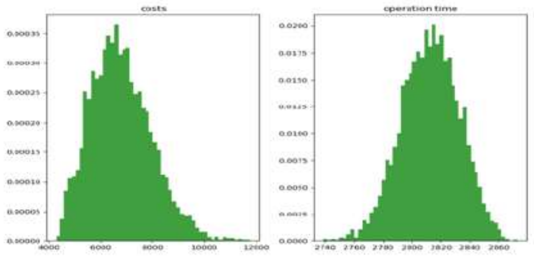


Figure 6. Histogram of distribution of costs and normal operation of the cycle during 5000 of test runs of the simulation

the distribution of the normal operations duration $F_{ir}(t_{wf})$ and probabilities of emergencies P_{ie} for nodes M_0, M_2, the ones for which the emergencies probabilities happens most frequently.

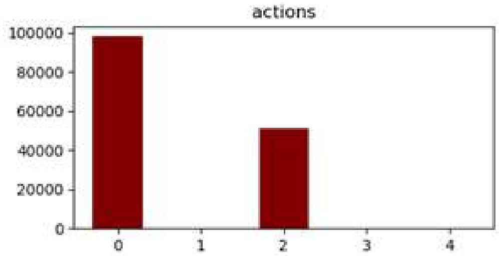


Figure 7. Distribution of the actions most frequently selected by the system

IX. Formalizing the controller as a problem solver of a decision making OSTIS system

In order to provide a possibility for the integration of the developed system concept with other intellectual systems a formalization of the proposed decision-making system based on OSTIS technology is proposed.

In the context of the OSTIS technology problem solvers are based on the multi-agent approach. According to this approach the problem-solver is implemented as a set of agents which are called sc-agents. These agents have shared memory and can exchange data through sc-texts. It is important to note that agents can be non-atomic, meaning that two or more sc-agents are operating to provide functionality for such an agent.

The problem solver for the task under consideration can be viewed as a decomposition of abstract non-atomic sc-agent.

abstract non-atomic sc-agent of cycle maintenance recommendation system
⇒ *decomposition of abstract sc-agent**:
{• *abstract sc-agent of interaction with the observation system*
• *abstract sc-agent of forming recommendations*
• *abstract sc-agent of forming maintenance requests*
}

1) abstract sc-agent of interaction with the observation system – performs extraction of observations from the means of hardware-software coupling in the technological production cycle, it initializes the operation of agent responsible for proposing recommendations.
2) abstract sc-agent of forming recommendations – based on the received observations initializes the operation of neurocontroller for receiving maintenance recommendations.
3) abstract sc-agent of forming maintenance requests – based on the data received from the agent of forming recommendation forms requests for maintenance for the corresponding means of hardware-software coupling.

X. Conclusion

In this paper an approach is proposed to constructing an intellectual system based on OSTIS technology for decision making when realizing the adaptive control procedures for the technological cycle.

The maintenance decision making system for the technological cycle is based on the neural network controller that is constructed using the methods of reinforcement learning for solving the task of optimal strategy search for the maintenance of the technological cycle.

A formalization of the decision making system based on the OSTIS technology is proposed, that allows integration into other intellectual systems when solving the task of technological production cycle control.

References

[1] V. Golenkov, N. Gulyakina, N. Grakova, I. Davydenko, V. Nikulenka, A. Eremeev, V. Tarasov. *From Training Intelligent Systems to Training Their Development Tools. Open Semantic Technologies for Intelligent Systems (OSTIS)*, Minsk, Belarussian State University of Informatics an Radioelectronics Publ., 2018, iss. 2, pp. 81–98.

[2] V. Golovko, A. Kroshchanka, V. Ivashenko, M. Kovalev, V. Taberko, D. Ivaniuk. *Principles of decision-making systems building based on the integration of neural networks and semantic models*, Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], 2019, pp. 91-102.

[3] Smorodin, V.S., Maximey, I.V. textitMethods and means of simulation modeling of technological production processes, Gomel, F. Skorina State University, 2007.

[4] Maximey, I.V., Smorodin, V.S., Demidenko, O.M. *Development of simulation models of complex technical systems*, Gomel, F. Skorina State University, 2014. 298 p.

[5] Sutton, R. S., Barto, A. G. *Reinforcement Learning: An Introduction*, Cambridge, The MIT Press, 1998./

[6] Sutton, R. S., McAllester, D., Singh S., Mansour Y. *Policy Gradient Methods for Reinforcement Learning with Function Approximation, Advances in Neural Information Processing Systems 1*, NIPS 1999./

Программно-технологический инструментарий адаптивного управления технологическим циклом роботизированного производства

Смородин В.С., Прохоренко В.А.

Предлагается подход к построению интеллектуальной системы нового поколения в рамках технологии OSTIS для принятия управляющих решений при реализации процедур адаптивного управления технологическим циклом роботизированного производства на базе средств программного-аппаратного сопряжения.

В основе интеллектуальной системы принятия решений лежит идея применения нейросетевых контроллеров, решающих задачи поиска оптимальной стратегии обслуживания технологического цикла роботизированного производства. Предлагается формализация подобной системы в рамках применения технологии OSTIS.

Received 16.11.2022

Towards semantic representation of the IoT ecosystem and smart home applications

Alexey Andrushevich, Iosif Vojteshenko
Belarusian State University
Minsk, Belarus
Email: andrushevich@bsu.by

***Abstract*—This paper describes an original approach to building a cross-industry ecosystem of the Internet of Things and smart home applications through its semantic representation based on OSTIS technology. The results will improve the efficiency of the component approach to application development in the Internet of Things in the future, as well as enable automatic synchronization of different versions of components, increasing their compatibility and consistency.**

***Keywords*—Internet of things, smart home, semantic representation, multi-component model**

I. Introduction

Multi-agent and situational (contextual) processing has found wide application in Internet of Things applications, such as the smart home [1]. However, despite the significant progress in recent years in the development of sectoral communication systems and automated control systems, the following problems are still relevant:

- heterogeneity of standards and technologies for receiving, storing, processing, and transmitting data in IoT applications;
- limited to the industry domain functionality;
- low adaptability, interoperability and scalability of applications.

Thus, it is necessary to state that there is no unified comprehensive approach to the design of an ecosystem of inter-industry universal Internet of Things (IoT).

II. Analysis of existing solution approaches

It has to be noted that the concept of a ubiquitous, cross-industry, context-aware, secure Internet of Things is rapidly gaining popularity due to the rapidly developing convergence of technology packages, system functions, platforms and services. Vivid examples of such convergence mechanisms are the emergence and development of common system functions in the Internet of Things, such as basic context awareness services / services, geolocation and information security. Different branches of IoT applications also often implement and use similar functional blocks, including user authentication and authorization, geo-temporal localization, event processing, contextual awareness and many others. For example, the scientific community has already published [2]–[4] the development results of underlying technologies and algorithms to acquire and provide information about geo-temporal location and context. In fact, the vision of a ubiquitous, pervasive Internet of Things includes not only a multitude of technologies, but also a dynamic changing environment and a virtual information environment (space) of users. The collection and use of contextual information goes beyond conventional closed-loop applications that use only simple location information. For example, universal geo-temporal information is successfully modernizing IoT applications in industries such as home, office, industry, commerce, transportation, healthcare, and cities.

III. Proposed approach

This paper proposes to take as a basis the well-known components and libraries of OSTIS [5] technology to formalize the description of the subject domain of the Internet of Things ecosystem. The systems developed on the basis of OSTIS Technology are called *ostis-systems*. At the heart of *OSTIS Technology* is a universal way of semantic representation (coding) of information in the memory of intelligent computer systems, called as *SC-code*. In this way, a better convergence of heterogeneous standards and IoT technologies can be achieved, which will contribute to the cross-industry integration of industry-specific communication systems and automated control systems into a single ecosystem.

So, let's give the following definitions of the considered subject area of the Internet of Things in *SC-code*:

Internet of Things

:= [A set of physical objects connected to the Internet and exchanging data. The term "Internet of Things" was first introduced in 1999 by Kevin Ashton, an entrepreneur and co-founder of Auto-ID Labs (a distributed research group in radio-frequency identification and new sensor technologies) at the Massachusetts Institute of Technology (MIT).]

:= [The concept of a data network between physical objects ("things") equipped with built-in means and technologies for interacting with each other or with the external environment]

Web of Things

:= [A global network of automatically generated web pages that are automatically read by computing devices embedded in physical objects (parking lots, paving tiles, windows, doors, children's toys, dishes, clothes, soil, etc.). The Web of Things is characterized by convergence and integration with artificial intelligence technologies]

:= [W3C's approach to using web technologies in the Internet of Things to eliminate fragmentation in Internet of Things development standards]

Context-dependent information system

:= [An information system that uses the concept of context to provide relationally personalized dynamic information and / or services to the user, where the degree of relationship depends on the user's current environment, interests, objectives, intentions, and actions]

API with representative state transfer

:= [The concept of building a distributed application based on the client-server type, where each request (REST-request) of the client to the server contains comprehensive information about the desired response (representative state) of the server without the need to store information about the client state (client session)]

Based on the authors' many years of professional experience, it can be argued that the organization of a universal generic cross-industry (or "horizontal") method of technological implementation of system architecture and applications in the Internet of Things can be represented as three development streams, each dedicated to the implementation of one system platform of the Internet of Things. These 3 fundamental application-independent platforms together form a generic IoT architectural platform that integrates into an interconnected heterogeneous system of IoT subsystems, offering common functionality to application developers and leaving developers more time to focus on the business logic of their applications. The *Cloud Server Platform* has high computing resources and large amounts of memory, manages the entire IoT system to the greatest extent. *Mobile Platform* is responsible for presenting the resulting and useful information to users. The *Connected Object Platform*, integrates sensory information about the environment and is capable of performing simple actions through connected objects.

Let us describe the above platforms in more details:

- Smart Server platform: a set of different servers to provide API building blocks in the cloud for integration of applications and services. This platform has high computational power, memory resources and can interface with any standards. One can consider the computational resources and capabilities as unlimited. Its role is to collect, aggregate and / or interpret context-sensitive information from connected IoT application nodes to make it efficient and meaningful (customer value) for mobile device users. It also provides application developers with APIs of local or remote universal core components that implement server-side functionality for all applications, such as: big data processing and analytics, complex event processing, a localization engine, a context management framework, a user profile and user behavior model, security policies, access control (including user registration, authentication and trust schema). The most common examples of Smart Server platform are Yandex.Cloud, Amazon Web Services (AWS), Microsoft Azure, Google Cloud, Cisco IoT Cloud Connect, SAP, Oracle IoT, ThingsBoard, SiteWhere, Predix IoT, Thinger.io;
- Smart Mobile platform: a reference platform for mobile applications that provides functionality on mobile devices for all IoT applications. Mobile devices are still most often owned by human end users. Common functions of mobile clients are: common user interface components, user profiling, authentication and authorization, information security (privacy), search and discovery of smart servers, communication with server components, receiving notifications from servers, "local" data mining algorithms (thick client). This platform provides IoT applications with the ability to use the contextual information they need without having to worry about how that contextual information is fetched. The mobile platform can communicate both with the Smart Server platform and directly with some connected IoT nodes. Finally, the mobile UI platform is responsible for the last stage of consumer value creation through the multimedia presentation / output of consistent and interesting information to users. The most common examples of the Smart Mobile platform are Zetta, HP Enterprise Universal, Carriots, ThingsBoard, ThingWorx, and Xively. To reduce the development time of mobile applications, so-called hybrid applications are often used, the code of which is adapted using frameworks for several mobile hardware platforms at once. The most popular frameworks are: PhoneGap, Rhodes, Appcelerator, Xamarin, Ionic, Appy Pie, Native Script.
- Smart Object platform: integration of various gateways, network hubs and related connected object technologies. Connected objects can perceive physical environmental data (temperature, pressure, light, humidity, vibration, etc.) through sensors or be actuators (switch, actuator, solenoid, etc.). Data transmission is usually organised through low-power communication standards. Such devices may also be equipped with a "lightweight" operating system.

They are very heterogeneous and can operate in very different ways. However, all implementation features of connected objects (things) must be hidden from other devices in order to provide a homogeneous way of communicating with other system components, regardless of what data they perceive or how they are arranged. Consequently, a single object software interface is a key to being able to easily integrate different gateways that provide access to different connected objects via the same standards, without requiring the designers of the UI application to know the complexity of the relationship between objects and the gateway and what functionality (data collection or actuation) is performed on those objects. This platform has common functions, such as: object detection, security, and management. The most common examples of the Smart Object platform are open source IoT Kaa, Arduino, Flutter, Qualcomm's IoT Development Kit, Particle.io, ESP8266, Intel Edison, Raspberry Pi, Beagle Bone.

IV. Multi-component model

This section is devoted to the theoretical definition of a universal multi-component model of a typical IoT application for the platforms from the previous subsection. A description for a "canonical" or "classic" Internet of Things application architecture consisting of cloud or server capacity, various data delivery protocols, user devices (PCs, laptops, tablets, phones, embedded user interfaces in any devices), any kind of data collection sensors and any "device-executor-hands". Taking into account the features and properties of the three parts of the horizontal platform (Smart Server, Smart Mobile and Smart Object), as well as the characteristics of Internet of Things applications, the following universal tree-like representation of a multi-component model of application architecture in the Internet of Things is proposed.

All the components and parameters of this hierarchical tree model are given as an example and are subject to change in specific IoT applications.

(0) Internet of Things application S = A * B * C

(1) Smart Server Platform A = D * E

(1.1) Access Control D = G * H * I

(1.1.1) Registration G: G1 (Users), G2 (Machines)

(1.1.2) Authentication H: H1 (Periodic), H2 (On Demand)

(1.1.3) Trust Scheme I: I1 (Application Managed), I2 (Policy Managed), I3 (Profile Managed), I4 (Risk Managed)

(1.2) Data Processing E = J * K * L

(1.2.1) Context management J: J1 (Event-driven), J2 (Polling)

(1.2.2) Flow processing K: K1 (Offline), K2 (Semi-Online), K3 (Online)

(1.2.3) Data storage L: L1 (Simple), L2 (Hierarchical), L3 (Structured), L4 (Dynamic)

(2) Smart Mobile Platform B = M * Phi

(2.1) General Mobility Characteristics M = O * P

(2.1.1) Basic data traffic type O: O1 (Multimedia), O2 (Perception / reading), O3 (Control), O4 (Well-balanced)

(2.1.2) Location information P: P1 (On / Off mode), P2 (Based on accuracy)

(2.2) User Interface Phi = R * F

(2.2.1) User Interface R: R1 (Single Media), R2 (Multimedia), R3 (Adaptive)

(2.2.2) Content delivery F: F1 (Service-based), F2 (Device-based)

(3) Smart Object Platform C = Q * T

(3.1) Common Node Characteristics Q = W * V * U

(3.1.1) Power Source W: W1 (Passive), W2 (Active)

(3.1.2) Physical Interaction V: V1 (Read), V2 (Action), V3 (Combined)

(3.1.3) Encryption U: U1 (Yes), U2 (No)

(3.2) Connectivity T = X * Y * Z

(3.2.1) Media X: X1 (Wireless), X2 (Wired)

(3.2.2) Network type Y: Y1 (Grid), Y2 (Point-to-Point)

(3.2.3) Configuration Z: Z1 (Adaptive), Z2 (Static)

Based on the proposed multi-component model of the Internet of Things ecosystem, such Internet of Things applications as smart home, office, industry, trade, transport, healthcare, and smart cities can be implemented.

Let us focus on the popular smart home application and describe its functions in more detail.

V. Examples of functional smart home tasks

Typical functional tasks implemented in the form of components / applications in the design and development of smart home software and hardware are [6]:

- task of access to the living quarters;
- task of monitoring lonely elderly people [7];
- task of controlling the lighting;
- task of energy management and energy efficiency [8].

Thus, the functional classification of smart home applications can be defined using the SC code as follows:

Smart home application
:= [separate hardware and software combination that operates according to functional requirements]
⇒ *partitioning**:
Partitioning a class of smart home applications by functionality
= {• *physical access control application*
⊃ *monitor / control the status of all entrances and exits*
• *supervisory application for elderly*
⊃ *domestic safety for elderly*
• *housing light control application*
⊃ *monitor and control the natural and artificial light*
• *energy management and energy efficiency application*
⊃ *monitor and control all resources and energy*
}

Let us describe in more details the functionality of the aforementioned smart home applications.

A. Housing access system

The key task of such a system is the identification of people who come near to the house entrance door and the correct processing of identification results: if the person who came to the door should have access to the house, the door opens automatically, in the opposite case the system offers to talk to the occupants of the house or apartment. In this case, it is desirable that the system is able to determine whether there is someone in the house. If no one is at home, the system must inform the visitor that he can not come in now.

In terms of devices, the system is equipped with a camera as well as lamps for lighting. The camera is triggered by the motion sensor, and the lights are also triggered if the environment is too dark for the camera to work. The system tries to recognize the visitor from the camera images.

In addition to the camera data, the system also has the residents' location data at its disposal. If no one is home according to the geo-location data, the system rejects visitors. There is also an option for the user to enable this mode manually. This can be useful, for example, if residents are not to be disturbed for a period of time.

The system should also contain a graphical user interface, through which one can both monitor selected indicators and the status of devices, and control the behavior of the system. In addition, through the user interface, residents can be notified when someone tries to enter the premises.

Thus, the following functional requirements for the system can be identified:

1) The system allows to set multiple resident profiles so that they can be identified on entrance.
2) If the identification is positive, the system opens the door automatically.
3) In the default state, the camera and lights are turned off, but they must be turned on as needed.
4) The system can determine whether or not anyone is in the house. If no one is home, the system should inform the visitor. If not, the system should offer to talk to people inside the house.
5) The system must also support the "Do Not Disturb" mode. The behavior of the system in this case corresponds to the case when no one is in the room.
6) The system must notify residents about visitors.

In addition to the functional requirements, the following non-functional requirements were also developed for the system:

1) The system can recognize up to 10 different resident profiles.
2) The system must respond correctly to device signals, even if there is a break in the connection.
3) Manual operation is also allowed, in particular the use of a key to open the door.
4) System extension with more devices is supported.

B. Surveillance System for Elderly People Living Alone

With the declining birth rate, the proportion of the elderly in society is increasing, while the proportion of people of working age is decreasing. Elderly people often need help with health care, free movement and in case of unwanted accidents. Through the use of Internet of Things technology, such people could live in greater safety and comfort. The field of the Internet of Things related to elderly supervision is also called AAL (ambient or active assisted living).

In [7] the following areas of application of IoT technologies were identified:

1. information assistance (easy accessibility of all necessary information),

2. intelligent situational behavior (the environment should recognize typical patterns of behavior and offer appropriate assistance),

3. prediction of undesirable events (recognition of such situations on the basis of behavioral and physiological indicators, and application of preventive measures),

4. recognizing and reacting to undesirable situations,

5. security (protection against intrusions using authorization and authentication mechanisms),

6. confidentiality (minimization of privacy interference).

To meet these needs, the system can rely on both historical and real-time data. Two groups of sensors are distinguished. Environmental data comes from environmental sensors, e.g., temperature, motion. Data on human behavior and condition comes from wearable sensors. The system may also include feedback devices for visual and voice notification of various events. State-of-the-art wireless technology allows for a reliable, easy-to-install, and inexpensive data communications infrastructure.

Central to AAL systems is the collection and storage of data on user behavior, highlighting patterns and identifying undesirable situations based on deviations from them. The user's state can be determined based on data from several sensors located in the dwelling [7].

Among the undesirable situations, let's especially emphasize falls. One way to detect falls is to use a wearable acceleration and atmospheric pressure sensor together. In the case of unexpected acceleration, the system reads the pressure data twice, on the basis of which it draws a conclusion about the position of the human body [7].

In terms of modeling such systems, three categories of indicators can be distinguished. The physical aspects include both environmental conditions and human health parameters. Due to the continuous nature of this category, the method of system dynamics combined with stochastic modeling to account for the role of randomness seems most natural. On the other hand, discrete-event modeling allows one to account for emerging events, such as falls. Finally, spontaneous user behavior is best modeled using the agent-based method.

C. System for dwelling light control

Approximate functional requirements for the application: When entering a room, the light comes on in response to movement and 10 seconds after leaving the room, the light turns off. The brightness of the light must be adjusted to the level of street light entering the dwelling. In addition, lighting from 11 p.m. to 6 a.m. shall operate in nightlight mode, with reduced brightness.

D. Energy consumption and efficiency management

Due to the growing energy crisis and the inadequacy of traditional centralized energy systems to new challenges, a new model has emerged: hybrid distributed energy production with a significant contribution of renewable energy sources. This model is also characterized by a bidirectional flow of information and electricity. There are solutions for all stages of energy production, but on the consumer side, the mainstream is the Internet of Things, in particular home automation. Such systems are called HEMS (home energy management system) [6]

Energy management systems have the following requirements [8]:

1) The system collects real-time data on energy consumption and production, as well as the status of devices.
2) The system stores and analyzes historical data.
3) The system manages the devices to ensure optimal energy consumption.
4) The user can control the devices directly and remotely.
5) The system warns the user in case of undesirable situations.

Let's consider the choice of metrics to evaluate the performance of energy management systems.

The environment in which energy management systems operate - residential premises, households, even office or industrial buildings - leads to the multipurpose nature of such systems [8], that is, the need to find a compromise between several objectives. While the global goal is always to increase energy efficiency, there are constraints on its achievement, in particular on user comfort. In addition, the formulation of a specific goal can vary depending on the context: emissions reduction, monetary savings, balancing the load on the energy system are some possible directions.

Energy management objectives can be expressed through cost. While the most obvious and predominant factor is the price of energy consumed, factors such as the initial installation costs of the system, the penalty for contributing to the total load, the projected depreciation of equipment, and the [9] greenhouse gas emissions tax can also be included in the overall calculation formula. In this way, the system can follow the single goal of reducing the monetary costs derived from this formula. Choosing the correct ratio can be difficult if there are no established monetary values for some goals.

The main objective of those that are difficult to represent in monetary terms is the comfort of users, that is, their inconvenience associated with the quality of service provided in the delivery of energy. The system should not lead to a significant change in their lifestyle. Various penalty functions can be used to assess the impact of the system on users' comfort: value limits, deviation, no-service penalty [9]. In other words, if cost represents some ratio to be minimized, then comfort requirements impose constraints on possible strategies.

The described above considerations can be useful to testing general approaches in optimizing energy consumption. For example, the tasks to estimate the benefits of adding a particular device and compare them with the purchase and installation costs, or to select the best algorithm among several algorithms under development.

From the end user's point of view, the system may have requirements such as access to historical data and trends, the ability to remotely manage devices, and warnings about undesirable situations [8]. Data privacy, reliability, and system efficiency also play a role in evaluating the quality of the system. Such requirements are more typical for practical products intended for direct intended use rather than research prototypes.

VI. Technical implementation details

Design and software implementation of the above components / applications was performed using the Node-RED visual programming tool combined with the use of cloud technologies, in particular Yandex IoT Core and AWS IoT Core. Node-RED is a flow programming tool for connecting hardware devices, APIs and online services. Their combined use allows prototyping Internet of Things systems without using real devices, allowing to focus through the architecture of the system before its physical implementation. One of the important advantages of Node-RED is that the tool can be used to create a simple prototype system in the same environment in which the main development is or will be carried out. Such an approach makes development much easier. Moreover, Node-RED also provides the means for easy visualization of the resulting system, which allows to prototype a graphical interface within the same environment.

The use of cloud services by applications, which include Yandex IoT Core and AWS IoT Core, significantly reduces system infrastructure costs while providing better scalability and fault tolerance. Cloud technologies offer computing, storage and communication resources.

For message transfer within the system we have used MQTT - a network protocol that uses a publisher-subscriber architecture and runs on top of TCP/IP using queuing (Message Queuing Telemetry Transport). It is convenient when used with low network bandwidth.

The implementation of a prototype system to control the lighting was carried out as described [10].

A. Running Node-RED on cloud-based virtual machine

To run the Node-RED environment, a virtual machine is created in the Yandex Compute Cloud service. This service is part of the Yandex Cloud and provides scalable computing power for creating and managing virtual machines. This service offers a wide range of virtual machine settings, from different operating systems to fine-tuning the resources used.

We have used a virtual machine based on the CentOS 8 operating system. As Node-RED didn't require much computing power to work with and execute the application prototype, the resources allocated to the virtual machine were minimal.

SSH was used to access the virtual machine. For this purpose, a public IP address was allocated in the settings of Yandex cloud services. SSH access to the virtual machine is necessary to install Node-RED, but the development can be carried out in a normal web browser. After generating a key pair for SSH and setting the corresponding public key in the virtual machine access settings, the virtual machine can be accessed using the command line interface. The official resource "Linux Installers for Node-RED" was used to install Node-RED. Node-RED was also enabled to start automatically when the virtual machine was started. Access to Node-RED was set to port 1880.

B. Integration of cloud service objects with Node-RED

The main elements of the Yandex IoT Core service are the device and the registry. These objects can exchange data and commands via MQTT protocol. The device in this service is an abstraction of a physical device, and the registry is a group of devices that are logically connected to each other. Data can be transferred between the device and the registry.

To add devices, one must first create a registry. To access devices and registries, IoT Core suggests setting a password, but one can also add a certificate. For our system, let's create one registry and three devices: a light, a camera, and a geolocation module. The latter is only needed for prototyping purposes and is an abstraction for real devices. It is not necessary to create sensors in the prototype, as this part is implemented through Node-RED.

To exchange MQTT messages between devices and registries, the IoT Core provides a MQTT broker, which is responsible for receiving, processing and delivering messages. Node-RED maintains a connection to the MQTT broker through dedicated MQTT Input and MQTT Output nodes. Usually, registries correspond to MQTT Input nodes, since this node allows to subscribe to messages of any topic and thus its data can be sent for processing. Devices more often correspond to MQTT Output nodes. To connect these nodes to IoT Core objects, one needs to specify identifiers, as well as to provide passwords. Finally, MQTT nodes need to be signed to the right topics in order to pass messages correctly.

VII. Conclusion

This paper describes an approach to describing the domain in IoT applications based on the example of a smart home based on OSTIS Technology.

The obtained results will improve in the future the efficiency of the component approach to the development of applications in the Internet of Things, as well as enable automatic synchronization of different versions of components, increasing compatibility and consistency.

Acknowledgment

The authors thank the teams of the Department of Software Engineering of the Belarusian State University and the Department of Intellectual Information Technologies of the Belarusian State University of Informatics and Radioelectronics for their help and valuable comments.

References

[1] A. Andrushevich, M. Staub, R. Kistler, and A. Klapproth, "Towards semantic buildings: Goal-driven approach for building automation service allocation and control," in *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*. IEEE, pp. 1–6.

[2] A. De Paola, P. Ferraro, S. Gaglio, and G. Lo Re, "Context-awareness for multi-sensor data fusion in smart environments," vol. 10037, pp. 377–391.

[3] A. Kamilaris and F. O. Ostermann, "Geospatial analysis and the internet of things," vol. 7, no. 7, p. 269, publisher: Multidisciplinary Digital Publishing Institute.

[4] R. Dobrescu, D. Merezeanu, and S. Mocanu, "Context-aware control and monitoring system with IoT and cloud support," vol. 160, pp. 91–99, publisher: Elsevier.

[5] V. Golenkov, N. Guliakina, and D. Shunkevich, "Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems," p. 690, 2021.

[6] B. R. Stojkoska and K. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[7] M. Biallas, E. Birrer, D. Bolliger, A. Rumsch, R. Kistler, A. Klapproth, and A. Andrushevich, "Living safely and actively in and around the home: Four applied examples from avatars and ambient cubes to active walkers," in *Safe at Home with Assistive Technology*. Springer, pp. 5–30.

[8] B. Zhou, W. Li, K. W. Chan, Y. Cao, Y. Kuang, X. Liu, and X. Wang, "Smart home energy management systems: Concept, configurations, and scheduling strategies," *Renewable and Sustainable Energy Reviews*, vol. 61, no. C, pp. 30–40, 2016. [Online]. Available: https://ideas.repec.org/a/eee/rensus/v61y2016icp30-40.html

[9] M. Beaudin and H. Zareipour, "Home energy management systems: A review of modelling and complexity," *Renewable and Sustainable Energy Reviews*, vol. 45, no. C, pp. 318–335, 2015. [Online]. Available: https://ideas.repec.org/a/eee/rensus/v45y2015icp318-335.html

[10] (2022, Oct) Programmiruem upravlenie osveshcheniem po datchikam dvizheniya i osveshcheniya na node-red [we program lighting control by motion and lighting sensors on node-red]. [Online]. Available: https://habr.com/ru/post/396985/

О семантическом представлении экосистемы интернета вещей и приложений умного дома

Андрушевич А. А., Войтешенко И. С.

В работе рассмотрены семантическое представление экосистемы интернета вещей и приложений умного дома на базе технологии OSTIS. Уточнена формальная трактовка таких понятий как интернет вещей, Веб Вещей.

Полученные результаты позволят повысить эффективность компонентного подхода к разработке приложений в интернете вещей, а также обеспечить возможность автоматической синхронизации различных версий компонентов, повышая совместимость и согласованность.

Received 31.10.2022

Next-generation intelligent geoinformation systems

Sergei Samodumkin
Belarusian State University of
Informatics and Radioelectronics
Minsk, Belarus
Email: Samodumkin@bsuir.by

***Abstract*—In the article, an approach to the building of intelligent geoinformation systems based on the OSTIS Technology is considered. The formal ontology of the syntax of the map language is explicitly set, which, in turn, allows establishing the types of map objects and setting spatial semantic relations; the formal ontology of the denotation semantics of the map language is set, which, in turn, allows establishing the semantics of displaying geo-entities on maps depending on the types of terrain objects; the formal ontology of terrain objects is set as a necessary condition for integration with subject domains in interests of GIS.**

***Keywords*—OSTIS, intelligent geoinformation system, private design technology, ontology**

I. Introduction

In geoinformatics, fundamental knowledge about space, time, and the Earth is systematically organized on the basis of information encoding.

The problems solved by geoinformation systems are directly related to inventory, analysis, modeling, prediction, and management of the environment and territorial organization of society and are inherently intelligent, i.e. the solution of which is included in the subject of the research of intelligent systems. At the same time, the nomenclature of problems involves quite independent subject domains, for example, the organization of transport [1], dispatching and ensuring the safety of spatial processes [2], geology [3], [4], energetics [5], [6].

At the dawn of the origination of geoinformation systems, development groups independently developed formats for storing spatial data, display tools, as well as sets of cartographic materials for the corresponding area. The situation has changed radically after the creation of web applications built on the basis of cartographic services and technologies provided by Google (Google Maps product) and Yandex (Yandex.Maps), as well as the development of the OpenMapStreet [7] project, directed on obtaining and providing open source geographic information. Thus, it became possible for third-party applications to repeatedly access distributed sets of cartographic data and, accordingly, obtain metric and semantic characteristics of terrain objects, i.e. to organize the representation, storage, and usage of geospatial entities.

At the same time, the lack of a single unified method of encoding information for solving GIS problems has led to the fact that for various directions of GIS application, their own models are being developed, adapted to the applied subject domain and the feature of spatial data organization.

In particular, a team of developers led by L. Massel et al. [5], [6] proposed a methodological approach to the integration of Earth's remote sensing (ERS) data based on data and knowledge integration methods in systematic energy research. For this purpose, the authors have developed a theoretical model of hybrid data based on a fractal stratified model (FS-model) of the information space. The hybrid data model is based on developing a system of ontologies of the ERS information space, including a meta-ontology describing the layers of the FS-model and ontologies of certain layers (subject domains).

As a result of ontological modeling, an ontological space is created, including a set of ontologies, which should allow working not only with data but also with knowledge, including descriptions of scenarios of various situations, models, and software complexes, and integrating them into the IT infrastructure of interdisciplinary research.

In the work [4], it is proposed to allocate geoconcepts for the classification of geospatial entities and the development of geo-ontologies of subject domains, that are in the sphere of interests of GIS users, to solve problems in geology.

In order to expand the problems solved by geoinformation systems, to unify various types of information representation in GIS about space, time, and the Earth, it is necessary to integrate existing web geoservices and intelligent systems design technologies in order to design next-generation geoinformation systems as a class of intelligent computer systems based on a unified way of encoding information and interoperability (compatibility) which is a necessary requirement.

Within the OSTIS Technology, powerful tools have been developed that allow describing any kind of knowledge in a unified form, structuring the knowledge base according to various criteria, as well as verifying its quality and editing the knowledge base directly during its operation [8], [9]. The basis of the knowledge base built on the OSTIS Technology is a hierarchical system of subject domains and their corresponding ontologies. The ontology is interpreted as a specification for the system of concepts of the corresponding subject domain, while various types of ontologies are distinguished, each of which reflects a certain set of properties for the concepts of the subject domain, for example, *terminological ontology*, *logical ontology*, *set-theoretic ontology*, etc.

II. Proposed approach

Within this article, it is proposed to take as a basis the approaches developed within the OSTIS Technology for the development of ontologies of subject domains and propose a hybrid knowledge model that ensures the integration of data and knowledge of the subject domains in geoinformatics, that is, propose a private technology for designing intelligent geoinformation systems, formally clarify the description of geo-entities, denotational semantics of geo-entities, propose basic mechanisms for processing geo-entities, and integrate a cartographic interface to provide a dialog with the user based on the language of questions.

The systems developed on the basis of the OSTIS Technology are called *ostis-systems*. The *OSTIS Technology* is based on a universal method of semantic representation (encoding) of information in the memory of intelligent computer systems, called an *SC-code*. Texts of the *SC-code* (sc-texts, sc-constructions) are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, depending on orientation, can be *sc-arcs* or *sc-edges*). The *Alphabet of the SC-code* consists of five main elements, on the basis of which SC-code constructions of any complexity are built, including more specific types of sc-elements (for example, new concepts). Memory that stores the SC-code constructions is called semantic memory, or *sc-memory*.

As it was mentioned earlier, the basis of the knowledge base within the OSTIS Technology is a hierarchical system of subject domains and ontologies. From there, to solve the problems set within this article, it is proposed to develop a complex *Subject domain of geoinformatics and the corresponding ontology of terrain objects*.

The development of the specified family of sc-models of the subject domains in geoinformatics, as well as geo-ontologies, will allow:

- describing geo-entities explicitly, which, in turn, allows determining the types of map objects and describing the geosemantic elements characteristic to them: location, topology, proximity, orientation, dynamics;
- establishing a formal ontology of the denotational semantics of the map language, which, in turn, will allow establishing the semantics of displaying geo-entities on maps depending on the types of terrain objects;
- establishing a formal ontology of terrain objects as a necessary condition for integration with subject domains in interests of GIS;
- creating tools for analyzing (understanding maps) and translating them into the internal language of knowledge bases, which will provide an understanding of the cartographic information stored in geoservices in relation to a specific subject domain;
- forming in the future a *kernel* of intelligent geoinformation systems and a *library* of components of intelligent geoinformation systems, which will allow the usage of the designed components for the development of applied geoinformation systems.

Next, we will consider in more detail the fragments of sc-models and ontologies for the development of intelligent geoinformation systems.

III. Stratified model of the information space of terrain objects

In order to integrate subject domains with spatial components of geoinformation systems and, respectively, to increase the interoperability of components of intelligent systems, a hybrid knowledge model is proposed. By this model we will understand a stratified model of the information space of terrain objects, which is formally defined in semantic memory as follows:

$$S^{\mu}, \mu \in I = \{S_{PO\mu}, S_{OM}, E_{OM}\}, \tag{1}$$

where I is a set of subject domains;

$S_{PO\mu}$ is an ontology of the μ -th subject domain;

S_{OM} is an ontology of terrain objects;

E_{OM} is instances of terrain objects.

In Figure 1, a geometric interpretation of the proposed hybrid model is demonstrated, where it is shown that the layer of instances of terrain objects is an integrating layer with subject knowledge of various subject domains in which specific terrain objects are already directly used. With such an organization of knowledge, it is possible to repeatedly use the developed ontology of terrain objects in different subject domain and, accordingly, to solve different applied problems.

IV. Formal description of geo-entities

In order to formally describe map objects, it is necessary to allocate the semantic properties of geo-entities. By the type of localization of map objects, areal, linear, multilinear, and point objects can be distinguished.

terrain object
=*cartographic object*
=*map object*
<= *subdividing**:

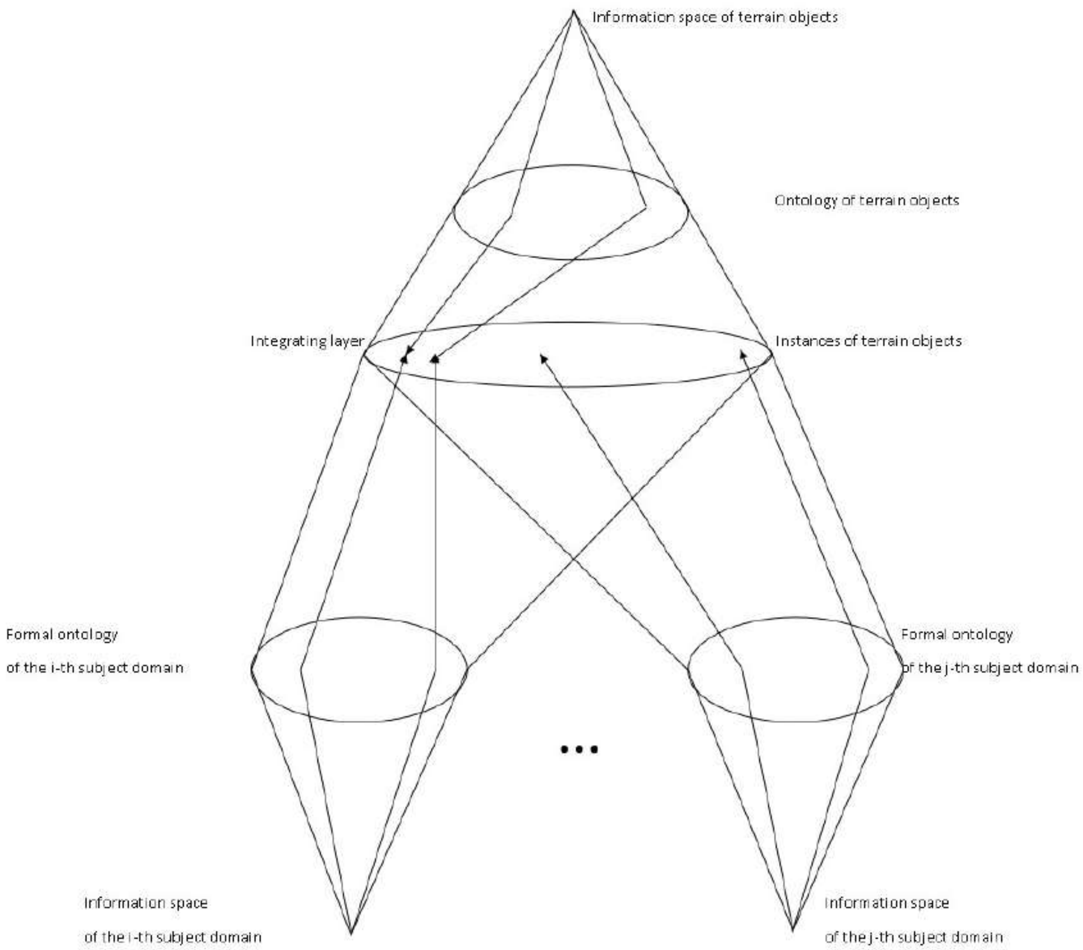

Information space of terrain objects
Ontology of terrain objects
Integrating layer
Instances of terrain objects
Formal ontology
of the i-th subject domain
Formal ontology
of the j-th subject domain
...
Information space
of the i-th subject domain
Information space
of the j-th subject domain

Figure 1. The stratified model of the information space of terrain objects

=*Subdividing by localization*
{
- *areal object*
- *linear object*
- *multilinear object*
- *point object*

}

Point objects are objects that cannot be expressed at the map scale. Linear and multilinear objects are objects whose length is expressed in a map scale, areal objects are objects whose area is expressed in a map scale. Accordingly, basic spatial semantic relations (SSR) are introduced over map objects, which, in turn, can be classified into three categories:

- cartographic (topological) relations invariant under topological transformations of relation objects;
- metric relations in terms of distance and direction;
- relations of spatial regularity described by prepositions before, behind, above, and below.

Accordingly, the following cartographic (topological) relations are introduced to establish spatial-logical connections.

cartographic relation
=*topological relation*
∋ *inclusion**
∋ *border**
∋ *intersection**
∋ *adjunction**

Thus, cartographic relations can be established between the above localities of terrain objects: *inclusion*, *border*, *intersection*, and *adjunction*.

V. Formal ontology of denotational semantics of the map language

In order to interpret the syntactic representation of objects on the map and establish the relation of connectivity and adjacency between objects on the map, it is necessary to clarify the denotational semantics of the map language, i.e. to indicate their semantics of display to the corresponding syntactic constructions of the map. So, for example, the topological relation of adjunction set over linear objects, depending on the type of the map object, can be interpreted as the adjunction of roads or the outlet of a river. In turn, for road map objects, topological relations, both adjunction and intersection, can be set with the corresponding denotation semantics, intersection or adjunction of roads, whereas for river map objects, only a topological relation of adjunction with denotation semantics, the outlet of one river into another, is possible.

VI. Formal ontology of terrain objects

The basis for building an ontological model of terrain objects is the classifier of topographic information displayed on topographic maps and city plans, developed and currently operating in the Republic of Belarus, NCRB 012-2007 [10]. In accordance with this condition, the objects of classification are terrain objects to which the map objects correspond, as well as the signs (characteristics) of these objects.

Let us set the subdivision of terrain objects on orthogonal bases, which corresponds to the location of objects in accordance with thematic layers in GIS.

terrain object
<= *subdividing**:
=*Subdividing by object type*
{
• *water objects and hydraulic facilities*
• *human settlements*
• *industrial, agricultural, and socio-cultural objects*
• *road network and road buildings*
• *vegetation cover and soils*
}

The ontology of terrain objects is a classification tree according to the hierarchy shown in Figure 2. Genus-species relations are set for each class of terrain objects.

For each terrain object, the main semantic characteristics inherent only to it are highlighted. It should be particularly noted that metric characteristics do not have such a property. According to this classifier, each class of terrain obejcts has a unique unambiguous denotation. The classifier hierarchy has eight classification levels and consists of a class code, subclass code, group code, subgroup code, team code, subteam code, species code, subspecies code. Thus, thanks to the coding method, genus-species relations have already been set, reflecting the relations of various classes of terrain objects, as well as the characteristics of a specific class of terrain objects have been established. Due to the fact that the basic properties and relations not of specific physical objects but their classes are set, such information is meta-information in relation to specific terrain objects, and the totality of this meta-information is an ontology of terrain objects, which, in turn, is part of the knowledge base of an intelligent geoinformation system.

As an example, we will demonstrate a fragment of the formal description of the "River Naroch" object in the knowledge base.

relation of spatial-logical connections between terrain objects*
:= [class of connections that characterize the spatial-logical relative position between terrain objects]
⇒ *first domain**:
terrain object
⇒ *second domain**:
terrain object
⊃ *inclusion of terrain object**
∈ *oriented relation*
∈ *binary relation*
⇒ *code**:
[91]
⇒ *requirement**:
[The internal and external contours of the same object should be connected logically.]
⊃ *belonging of terrain object**
⊂ *belonging**
⇒ *code**:
[205]
⇒ *requirement**:
[It is set for objects or their parts that are not connected metrically but have an explicit logical connection.]
⊃ *neighborhood of terrain objects**
∈ *non-oriented relation*
∈ *binary relation*
⇒ *code**:
[218]
⇒ *requirement**:
[It is set for the object and its caption (proper name, explanatory caption, characteristics caption).]
⊃ *intersection of terrain objects**
⊃ *intersection of sets**
⇒ *code**:
[298]
⇒ *requirement**:
[Intersecting terrain objects must have the same coordinates of the intersection point.]
⊃ *adjunction of terrain objects**
∈ *oriented relation*
∈ *binary relation*
⇒ *code**:
[298]

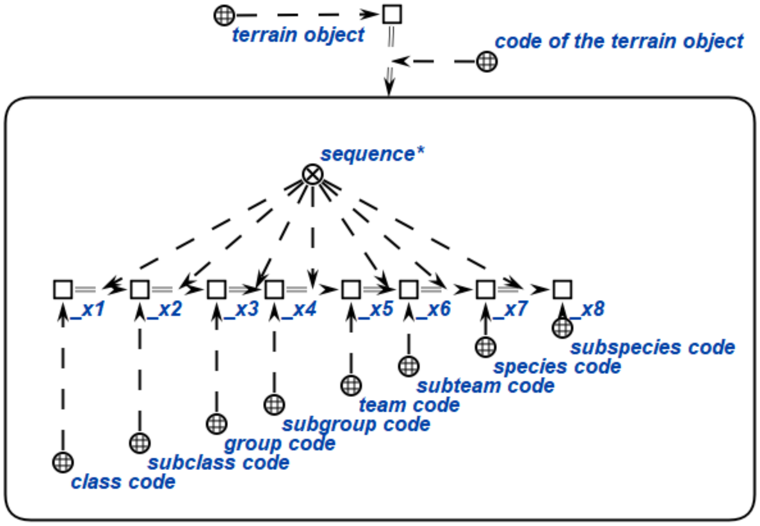


Figure 2. Hierarchy levels of classes of terrain objects

⇒ *requirement**:
[Adjacent terrain objects must have the same coordinates of the point at the junction.]

⊃ *extension of the terrain object**
∈ *oriented relation*
∈ *binary relation*
⇒ *code**:
[271]
⇒ *requirement**:
[The narrowed terrain objects must have common points on the frame of the nomenclature sheet (NS).]

terrain object parameter*
⊂ *parameter*
⊃ *relative height*
⇒ *code**:
[1]
⊃ *length*
⇒ *code**:
[2]
⊃ *absolute height*
⇒ *code**:
[4]
⊃ *depth*
⇒ *code**:
[7]
⊃ *distance*
⇒ *code**:
[24]

relation characterizing the property of the terrain object*
:= [class of relations that characterize a property or properties of a terrain object]
⇒ *first domain**:
terrain object
⊃ *vegetation type**
∈ *oriented relation*
∈ *binary relation*
⇒ *code**:
[62]
⊃ *border type**
∈ *oriented relation*
∈ *binary relation*
⇒ *code**:
[67]
⊃ *distributional pattern**
∈ *oriented relation*
∈ *binary relation*
⇒ *code**:
[78]

River Naroch
∈ *river*
⊂ *terrain object*
⇒ *feature codes**:
[4 5 9 31 33]
∈ *165*
∈ *absolute height*
⊂ *parameter*
⇒ *measurement on a meter scale**:
165 m
∈ *natural watercourse*
∈ *watercourse type*
⊂ *parameter*

∈ *natural watercourse*
  ∈ *watercourse type*
    ⊂ *parameter*
⇒ *proper name**:
  [Naroch**]**
⇒ *proximity period**:
  [May–September**]**
⇒ *water qualitative characteristics**:
  [High water quality is also characterized by indicators of the hydrochemical regime. The mineral content in the water does not exceed 250 mg/l with a low amount of chlorides and sulfates. High transparency is combined with low coloration (5–7 °). The indicator of organic matter – permanganate oxidizability – does not exceed 5-7 mgO/l. Biogenic elements – nitrogen and phosphorus – are also characterized by minimal values.**]**

VII. Conclusion

In the article, an approach to the building of intelligent geoinformation systems based on the OSTIS Technology is considered. The peculiarity of this approach is the description of geo-entities and the definition of spatial semantic relations, the description of the formal ontology of the denotation semantics in the map language, which, in turn, allows establishing the semantics of displaying geo-entities on maps depending on the types of terrain objects. Special attention is paid to the formal ontology of terrain objects as a necessary condition for ensuring integration with subject domains in interests of GIS.

At the next stage of technology development, the results obtained will make it possible to make tools for analyzing (understanding maps) and translating them into the internal language of knowledge bases, which, in general, will provide an understanding of the cartographic information stored in geoservices in relation to a specific subject domain, as well as further form a *kernel* of intelligent geoinformation systems and a *library* of components of intelligent geoinformation systems.

Acknowledgment

The author would like to thank the research group of the Departments of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for its help in the work and valuable comments.

The work was carried out with the partial financial support of the BRFFR (BRFFR-RFFR No. F20R-414).

References

[1] Belyakova, M. L., *Intellektual'nye geoinformacionnye sistemy dlya upravleniya infrastrukturoj transportnyh kompleksov [Intelligent geoinformation systems for managing the infrastructure of transport complexes]*. Taganrog : Southern Federal University Press, 2016, 190 p.

[2] Ivakin, Ya.A., “Metody intellektualizacii promyshlennyh geoinformacionnyh sistem na osnove ontologij [Methods of intellectualization of industrial geoinformation systems based on ontologies],” dis. ... doct. of techn. sc. : 05.13.06, Saint-Petersburg, 2009, 371 p.

[3] B. A.A., *Konceptual'noe proektirovanie GIS i upravlenie geoinformaciej. Tekhnologii integracii, kartograficheskogo predstavleniya, veb-poiska i rasprostraneniya geoinformacii [Conceptual GIS design and geoinformation management. Technologies of integration, cartographic representation, web search and distribution of geoinformation]*. LAP LAMBERT Academic Publishing, 2012, 484 p.

[4] B. A.A, “Semantika geoprostranstvennyh ob'ektov, funkcional'naya grammatika i intellektual'nye gis [Semantics of geospatial objects, functional grammar and intelligent GIS],” in *Proceedings of higher educational institutions. Geology and exploration*, no. 2, 2014, pp. 62–69.

[5] Massel, L. V., and Vorozhtsova, T. N., and Pyatkova, N. I., “Ontologicheskij inzhiniring dlya podderzhki prinyatiya strategicheskih reshenij v energetike [Ontological engineering to support strategic decision-making in the energy sector],” *Design Ontology*, vol. 7, no. 1(23), pp. 66–76, 2017.

[6] Mussel, L. V. and Kopaygorodsky, A. N., “Metodicheskij podhod k integracii dannyh distancionnogo zondirovaniya zemli na osnove metodov integracii dannyh i znanij v sistemnyh issledovaniyah energetiki [Methodological approach to the integration of earth remote sensing data based on data and knowledge integration methods in energy system research],” in *Spatial data processing and remote monitoring of the natural environment and large-scale anthropogenic processes (DRPS'2013) : Proceedings of the Conference, Barnaul, September 30 – October 04, 2013.*, Russian Academy of Sciences, Sib. department, In-t of Water and Environmental problems; editor: N.N. Dobretsov [et al.]. Barnaul, 2013, pp. 4–13.

[7] OpenStreetMap [OpenStreetMap]. [Online]. Available: http://www.prolog-plc.ru/

[8] Davydenko, I., “Semantic models, method and tools of knowledge bases coordinated development based on reusable components,” in *Otkrytye semanticheskie tehnologii proektirovanija intellektual'nyh sistem [Open semantic technologies for intelligent systems]*, V. Golenkov, Ed., BSUIR. Minsk , BSUIR, 2018, pp. 99–118.

[9] V. Golenkov, N. Guliakina, and D. Shunkevich, *Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems*, V. Golenkov, Ed. Minsk: Bestprint [Bestprint], 2021.

[10] “Cifrovye karty mestnosti; informaciya, otobrazhaemaya na topograficheskih kartah i planah naselennyh punktov : OKRB 012-2007 [digital terrain maps; information displayed on topographic maps and plans of human settlements: Ncrb 012-2007],” 2007.

Интеллектуальные геоинформационные системы нового поколения

Самодумкин С.А.

В работе рассмотрен подход к построению интеллектуальных геоинформационных систем на основе Технологии OSTIS. Явно задана формальная онтология синтаксиса языка карт, что, в свою очередь, позволило установить типы объектов карт и задать пространственные семантические отношения; задана формальная онтология денотационной семантики языка карт, что, в свою очередь, позволяет задать семантику отображения геосущностей на картах в зависимости от типов объектов местности; задана формальная онтология объектов местности как необходимое условие для интеграции с предметными областями в интересах ГИС.

Received 01.11.2022

Information security in intelligent semantic systems

Valery Chertkov
Euphrosyne Polotskaya State University of Polotsk
Polotsk, Belarus
Email: v.chertkov@psu.by

***Abstract*—The development of artificial intelligence causes the transition to semantic information processing technologies, which require the formation of new approaches to ensuring the information security systems. The article is devoted to the review of approaches and principles of ensuring security in intelligent systems of the new generation. The current state of ensuring information security in intelligent systems is given and the formed main goals and directions for the development of information security are presented. The methods of ensuring information security considered in the article are extremely important when analyzing the level of security of new generation intelligent systems.**

***Keywords*—information security, intelligent systems, semantic systems**

I. Introduction

One of the modern directions in the development of information technologies is the transition to working with the semantics of information and the creation of intelligent systems of a new generation [1]. The main advantage of which is the organized work with the semantic knowledge base. A feature of such a knowledge base is that the intelligent system is able to obtain new knowledge that is not directly contained in the database.

Since the design, construction and use of intelligent systems based on semantic knowledge bases began relatively recently, the issue of ensuring their security has not yet been fully resolved. In this regard, it is relevant to develop methods and algorithms that allow maintaining the safety of the functioning of such intelligent systems.

Currently, many methods have been developed to ensure information security in intelligent systems based on storing information in relational databases. But these methods cannot be used to ensure the security of semantic intelligent systems. Because such systems use semantic databases. Semantic databases are characterized by a strong hierarchical relationship between elements. Also, in semantic databases, it is possible to obtain new knowledge by using certain logical rules. Separate methods and algorithms have already been developed to solve the problems of ensuring the security of semantic databases: user access control based on named RDF graphs, user access control at the triplet level in the RDF storage. But the developed methods and algorithms have a number of shortcomings that do not allow to effectively ensure the comprehensive security of semantic databases.

II. Artificial intelligence and information security

Based on the analysis of literary sources, the information security of intelligent systems is considered from two aspects: 1) the use of artificial intelligence in information security; 2) organization of information security in intelligent systems.

A. Applications of artificial intelligence in information security

It should be noted that artificial intelligence (machine learning) is actively used to monitor and analyze security vulnerabilities in information transmission networks [2]. An artificial intelligence system allows machines to perform tasks more efficiently, such as visual perception, speech recognition, decision making, and translation from one language to another.

– intrusion detection: artificial intelligence can detect network attacks, malware infections and other cyber threats;

– cyber analytics: artificial intelligence is also used to analyze big data in order to identify patterns and anomalies in the organization's cyber security system;

– secure software development: native intelligence can help create more secure software by providing developers with real-time feedback.

In [3], a method for constructing a neuroimmune system for analyzing information security incidents is proposed, which combines modules for collecting and storing (compressing) data, a module for analyzing and correlating information security events, and a subsystem for detecting network attacks based on convolutional neural networks. The use of machine learning technologies in information security creates bottlenecks and system vulnerabilities that can be exploited and have the following disadvantages [4]:

- data sets that must be formed from a significant number of input samples, which requires a lot of time and resources;

– requires a huge amount of resources, including memory, data and computing power;

– frequent false positives that disrupt the operation and generally reduce the effectiveness of such systems;

– organized attacks based on artificial intelligence (semantic viruses).

B. Organization of information security in intelligent systems

Let's define the goals of ensuring the information security of new generation systems.

From the monograph by A.V. Ostroukh [5], the goals of ensuring the information security of traditional intelligent systems are to ensure the safety and confidentiality of information, protection and guarantee of the availability, reliability and integrity of information, avoiding information leakage, minimizing damage from events that threaten information security.

It should be noted that since the new generation of intelligent systems will interact with similar systems while understanding what the request is about, the goals of the provision will look different. The goals of ensuring the information security of new generation intelligent systems are: to ensure the safety of the semantic compatibility of information, to protect the reliability and integrity of information, to ensure the availability of information at different levels of the intelligent system, to minimize damage from events that threaten information security.

Currently, classical approaches and principles have been developed to ensure the security of knowledge bases (data), communication interfaces (information exchange) between the components of intelligent systems, such as encryption of transmitted data, filtering of unnecessary (redundant) content, and data access control policy.

The information security system should be created on the following principles:

- the principle of equal strength - means ensuring the protection of equipment, software and control systems from all types of threats;

- the principle of continuity - provides for continuous security of information resources, IP for the continuous provision of public services;

- the principle of reasonable sufficiency - means the application of such measures and means of protection that are reasonable, rational and the costs of which do not exceed the cost of violating information security;

- the principle of complexity - to ensure security in the whole variety of structural elements, threats and channels of unauthorized access, all types and forms of protection should be applied in full;

- the principle of comprehensive verification - is to conduct special studies and inspections, special engineering analysis of equipment, verification studies of software. Alarm messages and error parameters should be continuously monitored, hardware and software equipment should be constantly tested, as well as software integrity control, both during software loading and during operation;

- the principle of reliability - methods, means and forms of protection should reliably block all penetration routes and possible channels of information leakage;

- the principle of universality - security measures should block the paths of threats, regardless of the place of their possible impact;

- the principle of planning - planning should be carried out by developing detailed action plans to ensure the information security of all components of the system for the provision of public services;

- the principle of centralized management - within a certain structure, the organized and functional independence of the process of ensuring security in the provision of public services should be ensured;

- the principle of purposefulness - it is necessary to protect what must be protected in the interests of a specific goal;

- the principle of activity - protective measures to ensure security in the work of the process of providing services should be implemented with a sufficient degree of perseverance;

- the principle of qualification of service personnel - maintenance of equipment should be carried out by employees trained not only in the operation of equipment, but also in technical issues of ensuring the security of information;

– the principle of responsibility - the responsibility for ensuring information security must be clearly established, transferred to the appropriate personnel and approved by all participants as part of the information security process.

III. The main directions of ensuring information security of semantic intelligent systems

Consider the architecture of the Ecosystem OSTIS (Open Semantic Technology for Intelligent Systems), which is shown in Figure 1.

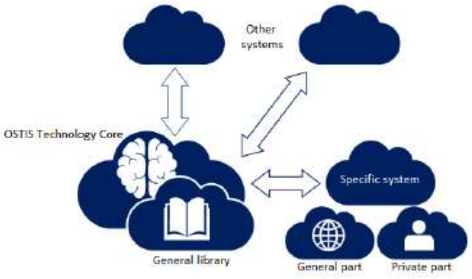


Figure 1. The architecture of the Ecosystem OSTIS.

The core of OSTIS technology includes: – OSTIS semantic knowledge base: it can describe any kind of knowledge, while it is easy to supplement it with new types of knowledge.

– OSTIS problem solver: Based on a multi-agent approach. This approach makes it easy to integrate and combine any problem solving models.

– interface of the OSTIS system: it is a subsystem with its own knowledge base and problem solver.

The presented architecture of the OSTIS Ecosystem implements:

– all knowledge bases are united into the Global Knowledge Base, the quality of which (logicality, correctness, integrity) is constantly checked by many agents. All problems are described in a single knowledge base, and specialists are involved to eliminate them, if necessary;

– each application associated with the OSTIS ecosystem has access to the latest version of all major OSTIS components, the components are updated automatically;

– each owner of the OSTIS Ecosystem application can share part of their knowledge for a fee or for free.

In the considered Ecosystem OSTIS, it is required to organize information security at each level: data exchange, data access rights, authentication of Ecosystem clients, data encryption, obtaining data from open sources, ensuring the reliability and integrity of stored and transmitted data, monitoring violation of links in the database knowledge. It should be noted that for some areas of ensuring the information security of semantic systems, methods and algorithms developed within the framework of traditional intelligent systems will be applied. For intelligent systems of the new generation, a number of aspects can be distinguished, within which the development of new algorithms and methods for ensuring information security is required. Let us present the main directions of ensuring the information security of semantic intelligent systems.

A. Restriction of information traffic analyzed by the intelligent system

The exponential growth of the amount of information circulating in information flows and resources under the conditions of quite definite quantitative restrictions on the capabilities of the means of its perception, storage, transmission and transformation forms a new class of information security threats characterized by the redundancy of the total incoming information traffic of intelligent systems.

As a result, the overflow of information resources of an intelligent system with redundant information can provoke the spread of distorted (destructive semantic) information. The general methodology for protecting intelligent systems from useless information is carried out through the use of axiological filters that implement the functions of numerical evaluation of the value of incoming information, selection of the most valuable and screening (filtering) of less valuable (useless or harmful) using well-defined criteria.

Active means of destroying the semantics of knowledge bases (semantic viruses) should also be singled out as a separate category of information security threats [6].

B. Knowledge base access control policy

Mandatory security policy (MAC - mandatory access control) is based on mandatory (forced) access control, which is determined by four conditions:

- all subjects and objects of the system are identified;

- a lattice of information security levels is specified;

- each object of the system is assigned a security level that determines the importance of the information contained in it;

- each subject of the system is assigned an access level that determines the level of trust in him in the intellectual system.

In addition, the mandate policy has a higher degree of reliability. The implementation of this policy is based on the developed algorithm for determining the agreed security levels for all elements of the ontology.

Since semantic knowledge bases, unlike a relational database, allow executing rules for obtaining logical conclusions, it is relevant to ensure data security by developing algorithms and methods that can only receive data that have security levels less than the access levels of the subjects who requested them [7].

1) Connectivity: All information stored in the semantic memory of the intelligent system is systematized in the form of a single knowledge base. Such information includes directly processed knowledge, interpreted programs, formulations of tasks to be solved, plans and protocols for solving problems, information about users, a description of the syntax and semantics of external languages, a description of the user interface, and much more [8]. In the information knowledge base between fragments of information (units of information), the possibility of establishing links of various types should be provided. First of all, these links can characterize the relationship between information units. Violation of connections leads to an incorrect logical conclusion, or to obtaining false knowledge, or to incompatibility of knowledge in the base.

2) Application of semantic metric: On a set of information units, in some cases it is useful to set a relation that characterizes the situational proximity of information units, i.e. the strength of the association between information units. It could be called the relevance relation for information units [9]. This attitude makes it possible to single out some typical situations in the knowledge base. The relevance relation when working with information units allows you to find knowledge that is close to what has already been found.

3) Semantic Compatibility: Internal semantic compatibility between the components of an intelligent computer system (i.e., the maximum possible introduction of common, coinciding concepts for various fragments of a stored knowledge base), which is a form of convergence and deep integration within an intelligent computer system for various types of knowledge and various problem solving models, which ensures effective implementation of the multimodality of an intelligent computer system. External semantic compatibility between different intelligent computer systems, which is expressed not only in the commonality of the concepts used, but also in the commonality of basic knowledge and is a necessary condition for ensuring a high level of socialization of intelligent computer systems [10].

4) Activity: In an intellectual system, the knowledge available in this system contributes to the actualization of certain actions. Thus, the execution of activities in an intelligent system should be initiated by the current state of the knowledge base. The appearance in the database of facts or descriptions of events, the establishment of links can become a source of system activity [11]. Including deliberate distortion of information and communications can become a source of deliberate distortion of information.

IV. Conclusion

Currently, there are no semantic knowledge bases in which internal interpretability, structuring, coherence would be fully implemented, a semantic measure would be introduced, and knowledge activity would be ensured. The methods of ensuring information security considered in the article are extremely important when analyzing the level of security of new generation intelligent systems. Systems that comply with the semantic security model will be resistant to attacks based on plain texts.

References

[1] V. V. Golenkov, N. A. Gulyakina, I. T. Davydenko, and D. V. Shunkevich, “Semantic technologies of intelligent systems design and semantic associative computers,” *Doklady BGUIR*, vol. 3, pp. 42–50, 2019.

[2] S. Isoboev, D. Vezarko, and A. Chechel’nitskii, “Intellektual’naya sistema monitoringa bezopasnosti seti besprovodnoi svyazi na osnove mashinnogo obucheniya,” *Ekonomika i kachestvo sistem svyazi*, vol. 1(23), pp. 44–48, 2022.

[3] V. A. Chastikova and A. I. Mityugov, “Metodika postroeniya sistemy analiza intsidentov informatsionnoi bezopasnosti na osnove neiroimmunnogo podkhoda,” *Elektronnyi Setevoi Politematicheskii Zhurnal «Nauchnye Trudy Kubgtu»*, vol. 1, pp. 98–105, 2022.

[4] D. D. Abdurakhman, “Iskusstvennyi intellekt i mashinnoe obuchenie v kiberbezopasnosti,” *Sovremennye problemy lingvistiki i metodiki prepodavaniya russkogo yazyka v vuze i shkole*, vol. 34, pp. 916–921, 2022.

[5] A. V. Ostroukh, *Intellektual’nye sistemy: monografiya*. Krasnoyarsk: Nauchno-innovatsionnyi tsentr, 2020.

[6] A. Palagin, “Semanticheskie aspekty informatsionnoi bezopasnosti: kontsentratsiya znanii,” *Istoriya i arkhivy*, vol. 13(75), pp. 38–58, 2011.

[7] K. Khoang and A. Tuzovskii, “Resheniya osnovnykh zadach v razrabotke programmy podderzhki bezopasnosti raboty s semanticheskimi bazami dannykh,” *Doklady TUSURa*, vol. 2(28), pp. 121–125, 2013.

[8] V. V. Golenkov, N. A. Gulyakina, I. T. Davydenko, and D. V. Shunkevich, “Semanticheskaya model’ predstavleniya i obrabotki baz znanii,” in *Data analytics and management in data-intensive fields: a collection of scientific papers of the XIX International Conference (DAMDID/RCDL’2017).* Moscow: Federal’nyi issledovatel’skii tsentr "Informatika i upravlenie" Rossiiskoi akademii nauk, 2017, pp. 412–419.

[9] A. V. Dement’ev, “Metriki semanticheskikh dannykh,” *Molodoi uchenyi*, vol. 24(419), pp. 48–51, 2022.

[10] V. V. Golenkov, N. A. Gulyakina, and D. V. Shunkevich, “Tekushchee sostoyanie i napravleniya razvitiya tekhnologii iskusstvennogo intellekta,” in *Informatsionnye tekhnologii i sistemy 2018 (ITS 2018): materialy mezhdunarodnoi nauchnoi konferentsii [Information Technologies and Systems 2018 (ITS 2018)].* Minsk : BSUIR, 2018, pp. 11–16.

[11] V. N.Druzhinina and D. V. Ushakova, *Kognitivnaya psikhologiya. Uchebnik dlya vuzov*. Moscow: PER SE, 1974.

Информационная безопасность интеллектуальных семантических систем

Чертков В. М.

Развитие искусственного интеллекта обуславливает переход на семантические технологии обработки информации, которые требует формирование новых подходов к обеспечению информационной безопасности таких систем. Статья посвящена обзору подходов и принципов обеспечения безопасности в интеллектуальных системах нового поколения. Приводиться современное состояние обеспечения информационной безопасности в интеллектуальных системах и представлены сформированные основные цели и направления по развитию обеспечения информационной безопасности. Рассмотренные в статье методы обеспечения безопасности информации являются чрезвычайно важными при анализе уровня защищённости интеллектуальных систем нового поколения.

Received 01.11.2022

AUTHOR INDEX

АВТОРСКИЙ УКАЗАТЕЛЬ

Научное издание

Открытые семантические технологии проектирования интеллектуальных систем

Open Semantic Technologies for Intelligent Systems

Сборник научных трудов

Основан в 2017 году

Выпуск 6

В авторской редакции

Ответственный за выпуск *В. В. Голенков*
Компьютерная вёрстка *Н. В. Гракова*

Подписано в печать 21.11.2022. Формат 60*84 1/8.
Бумага офсетная. Гарнитура «Таймс». Ризография.
Усл. печ. л. 50,22. Уч.-изд. л. 44,62.
Тираж 105 экз. Заказ 113.

Полиграфическое исполнение
УП «Бестпринт». Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий №1/160 от 27.01.2014.
Ул. Филатова, д. 9, к. 1, 220026, г. Минск

OSTIS-2023

12th international scientific and technical conference «Open Semantic Technologies for Intelligent Systems»

Open Semantic Technologies for Intelligent Systems

April 20-22, 2023 Minsk. Republic of Belarus

CALL FOR PAPERS

We invite you to take part in XIII International Scientific and Technical Conference “Open Semantic Technologies for Intelligent Systems” (OSTIS-2023), which will focus on areas of use of the semantic technologies.

Conference will take place from ***April, 20th to April, 22th, 2022*** at the Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus.

Research Papers Collection language: English

Working languages of the conference: Russian, Belarusian, English

MAIN ORGANIZERS OF THE CONFERENCE

- Ministry of Education
- Ministry of Communications and Informatization
- State Institution “Administration of High Technologies Park” (Republic of Belarus)
- Educational-scientific association in the direction of "Artificial Intelligence" (ESA-AI)
- Belarusian State University of Informatics and Radioelectronics (BSUIR)
- Brest State Technical University (BrSTU)
- The State Scientific Institution «The United Institute of Informatics Problems of the National Academy of Sciences of Belarus» (UIIP NASB)
- Russian Association of Artificial Intelligence (RAAI)
- Belarusian Public Association of Artificial Intelligence Specialists (BPA of Artificial Intelligence Specialists)

CONFERENCE TOPICS:

- *Underlying principles of semantics-based knowledge representation, and their unification.*
  Types of knowledge and peculiarities of the semantics-based representation of various knowledge and metaknowledge types.
  Links between knowledge; relations, that are defined on the knowledge.
  Semantic structure of a global knowledge base, that integrates various accumulated knowledge.
- *Parallel-oriented programming languages for processing of the semantics-based representation of knowledge bases.*
- *Models for problem solving, that are based on knowledge processing, which occurs directly at the semantics-based representation level of knowledge being processed. Semantic models of information retrieval, knowledge integration, correctness and quality analysis of knowledge bases, garbage collection, knowledge base optimization, deductive and inductive inference in knowledge bases, plausible reasoning, pattern recognition, intelligent control. Integration of various models for problem solving*
- *Semantic models of environment information perception and its translation into the knowledge base.*
- *Semantic models of multimodal user interfaces of intelligent systems, based on the semantic representation of knowledge used by them, and unification of such models.*
- *Semantic models of natural language user interfaces of intelligent systems. The structure of semantic representation of linguistic knowledge bases, which describe natural languages and facilitate solution of natural language text and speech interpretation problems, and of natural language texts and speech messages synthesis, that are semantically equal to certain knowledge base fragments.*
- *Integrated logic-semantic models of intelligent systems, based on semantic knowledge representation, and their unification*
- *Various technical platforms and implementation variants of unified logic-semantic models of intelligent systems, based on semantic knowledge representation*
- *Models and means, that are based on the semantic representation of knowledge and that are oriented to the design of various typical components of intelligent systems (knowledge bases, programs, problem solvers, user interfaces).*
- *Models and means, that are based on semantic representation of knowledge and that are oriented to the complex design of various classes of intelligent systems (intelligent reference systems, intelligent learning systems, intelligent control systems, intelligent robotics systems, intelligent systems for design support etc.)*
- *Applied intelligent systems, that are based on the semantic representation of knowledge used by them*

CONFERENCE GOALS AND FORMAT

The goal of the conference is to discuss problems of creation of the **Open Complex Semantic Technology for Hybrid Intelligent Systems Design**. This determines the Conference format, which involves wide discussion of various questions of creating of such technology and poster sessions.

During the **poster sessions** every participant of the conference will have an opportunity to demonstrate his results. Conference format assumes exact start time of each report, and exact time of its exhibition presentation.

One of the major objectives of the conference is to attract not only scientists and postgraduate students, but also students who are interested in artificial intelligence, as well as commercial organizations willing to collaborate with research groups working on the development of modern technologies for intelligent systems design.

PARTICIPATION TERMS AND CONDITIONS

All those interested in artificial intelligence problems, as well as commercial organizations willing to collaborate with research groups working on the development of modern technologies for intelligent systems design are invited to take part in the Conference.

To participate in the OSTIS-2023 conference, it is necessary to register in the CMT system **before March 13, 2023**, find conference page, and from there:

- submit a **participation form** for the OSTIS conference. Each participation form field is required, including indication of the reporter. By filling in the registration form, you agree that your personal data will be processed by the Organizing Committee of the Conference, and that the paper and information about the authors will be published in printed and electronic format. Participation form should contain information on all of the authors. If author(s) are participating with a report, participation form should have their **color photo(s)** attached (they are needed for the Conference Program);
- upload an **article** for publication in the OSTIS Research Papers Collection. Papers should be formatted according to the provided template (see https://proc.ostis.net/for-authors/). Four full pages is a minimum size of a paper.
- send the signed **scan of the letter of consent**

If a report is submitted to participate in one of the contests, this intention should be clearly indicated in the participation form.

The selection of papers for publication in the Research Papers Collection and participation in the Conference is performed by a number of reviewers from among the members of the Conference Program Committee.

Incompliant applications and papers will be rejected.

Conference participation does not require any fees.

PAPERS SUBMISSION PROCEDURE

Papers (only on topics mentioned above) should be submitted ready for publication (http://proc.ostis.net -> For authors). The text should be logically complete and contain new scientific and practical results. Each author is allowed to submit two reports maximum.

After receiving the article, it is sent for review. The authors can get acquainted with the results of the review in CMT, if necessary, correct the comments of the reviewers and send them for re-review.

The Organizing Committee reserves the right to reject any paper, if it does not meet the formatting requirements and the Conference topics, as well as if there was no participation form submitted for the paper.

YOUNG SCIENTIST REPORTS CONTEST

Authors of the report submitted to the contest may include scientists with scientific degrees, but the report should be made by those without a degree and under 35 years old.

To take part in the young scientists report contest, it is necessary to:

1) fill in the participation form, where your participation in the contest is clearly indicated;

2) write an article and upload it to the CMT website;

3) fill in, sign, scan and send letter of consent via the email.

4) make a report at the conference (in person);

YOUNG SCIENTIST PROJECTS CONTEST

Projects of applied intelligent systems and systems aimed at supporting the design of intelligent systems are allowed to take part in the contest; they have to be presented by a scientist without a degree and under 35 years old.

To take part in the young scientist projects contest, it is necessary to:

1) fill in the participation form, where your participation in the contest is clearly indicated;
2) write an article and upload it to the CMT website;
3) make a report at the conference (in person);
4) make an exhibition presentation of the software

STUDENT INTELLIGENT SYSTEM PROJECTS CONTEST

To participate in the contest, a project must meet the following criteria: (a) it was developed by students and/or undergraduates of the higher education institutions, and (b) project consultants and advisors must hold a scientific degree and title. To participate in this contest, it is necessary to:

1) familiarize yourself with contest's terms and conditions (http://conf.ostis.net);
2) fill in the participation form for the contest (http://conf.ostis.net);
3) prepare a summary of the project (http://conf.ostis.net).
4) submit the participation form and project summary to the student projects' email address: ostis.stud@gmail.com.

CONFERENCE PROCEEDINGS PUBLICATION

The Conference Organizing Committee plans to publish the papers selected by the Program Committee based on the results of their review, in the Conference Proceedings, on the official Conference website http://conf.ostis.net and on the Conference Proceedings website http://proc.ostis.net**.**

Upon successful review author sends a letter of consent to the Organizational Committee. Author therefore agrees that his paper can be made freely available in electronic form at other resources at the Editorial Board's discretion.

Since 2020, the OSTIS Research Papers Collection has been included in the List of Scientific Publications of the Republic of Belarus for publishing the results of dissertation research (List of the Higher Attestation Commission of the Republic of Belarus) in the technical field of science (informatics, computer technology and management).

In addition, following the results of the conference, it is planned to publish the Selected Papers Collection of the OSTIS conference in the series "Communications in Computer and Information Science" (CCIS) published by Springer. Detailed information about this can be found on the conference website (http://conf.ostis.net).

KEY DATES OF THE CONFERENCE

March 1, 2023	paper submission opens
March 13, 2023	paper submission deadline
March 27, 2023	paper review deadline
April 10, 2023	final decision on paper publication; sending out invitations and notifications on inclusion of a paper in the OSTIS Research Papers Collection
April 13, 2023	Draft Conference Program publication on the conference website http://conf.ostis.net
April 17, 2023	Research Papers Collection publication on the conference website http://proc.ostis.net
April 20, 2023	Participant registration and OSTIS-2023 conference opening
April 20-22, 2023	OSTIS-2023 Conference
May 2, 2023	Photoreport and conference report publication on the conference website: http://conf.ostis.net
May 20, 2023	Research Papers Collection will be uploaded to the Russian Science Citation Index database

CONFERENCE PROGRAM FORMATION

Conference program is formed by the Program Commitee according to the paper review results; author(s)' confirmation of participation is required as well.

CONTACTS

All the necessary information about the forthcoming and previous OSTIS Conferences can be found on the conference website http://conf.ostis.net and http://proc.ostis.net.

For questions regarding conference participation and dispute resolution please contact: ostisconf@gmail.com.

Methodological and advisory support to the conference participants shall be provided through the conference e-mail only.

The conference venue is the 5th academic building of the Belarusian State University of Informatics and Radioelectronics (39, Platonov str., Minsk, Republic of Belarus).

OSTIS-2023

XIII международная научно-техническая конференция «Открытые семантические технологии проектирования интеллектуальных систем»

Open Semantic Technologies for Intelligent Systems

20 – 22 апреля 2023 г. Минск. Республика Беларусь

ИНФОРМАЦИОННОЕ ПИСЬМО

Приглашаем принять участие в XIII Международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» (OSTIS-2023), которая будет посвящена вопросам области применения семантических технологий.

Конференция пройдет в период с ***20 по 22 апреля 2023*** года в Белорусском государственном университете информатики и радиоэлектроники, г. Минск, Республика Беларусь.

Язык статей сборника научных трудов: английский

Рабочие языки конференции: русский, белорусский, английский.

ОСНОВНЫЕ ОРГАНИЗАТОРЫ КОНФЕРЕНЦИИ

- Министерство образования Республики Беларусь
- Министерство связи и информатизации Республики Беларусь
- Государственное учреждение «Администрация Парка высоких технологий» (Республика Беларусь)
- Белорусский государственный университет информатики и радиоэлектроники (БГУИР)
- Брестский государственный технический университет (БрГТУ)
- Государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси» (ОИПИ НАН Беларуси)
- Белорусское общественное объединение специалистов в области искусственного интеллекта (БОИИ)
- Российская ассоциация искусственного интеллекта (РАИИ)
- Учебно-научное объединение по направлению «Искусственный интеллект» (УНО-ИИ)

НАПРАВЛЕНИЯ РАБОТЫ КОНФЕРЕНЦИИ:

- *Принципы, лежащие в основе семантического представления знаний, и их унификация.*
  Типология знаний и особенности семантического представления различного вида знаний и метазнаний.
  Связи между знаниями и отношения, заданные на множестве знаний.
  Семантическая структура глобальной базы знаний, интегрирующей различные накапливаемые знания
- *Языки программирования, ориентированные на параллельную обработку семантического представления баз знаний*
- *Модели решения задач, в основе которых лежит обработка знаний, осуществляемая непосредственно на уровне семантического представления обрабатываемых знаний. Семантические модели информационного поиска, интеграции знаний, анализа корректности и качества баз знаний, сборки информационного мусора, оптимизации баз знаний, дедуктивного и индуктивного вывода в базах знаний, правдоподобных рассуждений, распознавания образов, интеллектуального управления. Интеграция различных моделей решения задач*
- *Семантические модели восприятия информации о внешней среде и отображения этой информации в базу знаний*
- *Семантические модели мультимодальных пользовательских интерфейсов интеллектуальных систем, в основе которых лежит семантическое представление используемых ими знаний, и унификация этих моделей*
- *Семантические модели естественно-языковых пользовательских интерфейсов интеллектуальных систем. Структура семантического представления лингвистических баз знаний, описывающих естественные языки и обеспечивающих решение задач понимания естественно-языковых текстов и речевых сообщений, а также задач синтеза естественно-языковых текстов и речевых сообщений, семантически эквивалентных заданным фрагментам баз знаний*
- *Интегрированные комплексные логико-семантические модели интеллектуальных систем, основанные на семантическом представлении знаний, и их унификация*
- *Различные технические платформы и варианты реализации интерпретаторов унифицированных логико-семантических моделей интеллектуальных систем, основанных на семантическом представлении знаний*
- *Средства и методы, основанные на семантическом представлении знаний и ориентированные на проектирование различных типовых компонентов интеллектуальных систем (баз знаний, программ, решателей задач, интерфейсов)*
- *Средства и методы, основанные на семантическом представлении знаний и ориентированные на комплексное проектирование различных классов интеллектуальных систем (интеллектуальных справочных систем, интеллектуальных обучающих систем, интеллектуальных систем управления, интеллектуальных*

робототехнических систем, интеллектуальных систем поддержки проектирования и др.)

- *Прикладные интеллектуальные системы, основанные на семантическом представлении используемых ими знаний*

ЦЕЛЬ И ФОРМАТ ПРОВЕДЕНИЯ КОНФЕРЕНЦИИ

Целью конференции является обсуждение проблем создания **открытой комплексной семантической технологии компонентного проектирования семантически совместимых гибридных интеллектуальных систем**. Этим определяется и формат её проведения, предполагающий широкое обсуждение различных вопросов создания указанной технологии и выставочные презентации докладов.

Выставочная презентация докладов даёт возможность каждому докладчику продемонстрировать результаты своей разработки на выставке. Формат проведения конференции предполагает точное время начала каждого доклада и точное время его выставочной презентации.

Важнейшей задачей конференции является привлечение к её работе не только учёных и аспирантов, но и студенческой молодежи, интересующейся проблемами искусственного интеллекта, а также коммерческих организаций, готовых сотрудничать с научными коллективами, работающими над интеллектуальными системами и созданием современных технологий и их проектированием.

УСЛОВИЯ УЧАСТИЯ В КОНФЕРЕНЦИИ

В конференции имеют право участвовать все те, кто интересуется проблемами искусственного интеллекта, а также коммерческие организации, готовые сотрудничать с научными коллективами, работающими над созданием современных технологий проектирования интеллектуальных систем.

Для участия в конференции OSTIS-2023 необходимо **до 13 марта 2023** года зарегистрироваться в системе CMT, найти страницу конференции и на ней:

- подать **заявку** на конференцию OSTIS. Каждое поле заявки обязательно для заполнения, в том числе указание того автора, кто будет представлять доклад. Заполняя регистрационную форму, Вы подтверждаете согласие на обработку Оргкомитетом конференции персональных данных, публикацию статей и информации об авторах в печатном и электронном виде. В заявке должна содержаться информация по каждому автору;
- загрузить **статью** для публикации в Сборнике научных трудов конференции OSTIS. Статья должна быть оформлена в соответствии с правилами оформления публикуемых материалов и занимать не менее 4 полностью заполненных страниц;
- загрузить **сканированный вариант письма о согласии** на публикацию и размещения передаваемых материалов в сети Интернет;
- загрузить **цветные фотографии** всех авторов статьи (это необходимо для оформления Программы конференции)

Если доклад представляется на конкурс докладов молодых учёных или на конкурс программных продуктов молодых учёных, это должно быть явно указано в заявке статьи (в CMT).

Отбор статей для публикации в Сборнике и участия в работе конференции осуществляется рецензентами и редакционной коллегией сборника.

Заявки и статьи, оформленные без соблюдения предъявляемых требований, не рассматриваются.

Участие в конференции не предполагает организационного взноса.

ПОРЯДОК ПРЕДСТАВЛЕНИЯ НАУЧНЫХ СТАТЕЙ

Статьи (только по перечисленным выше направлениям) представляются в готовом для публикации виде (*http://proc.ostis.net -> Авторам*). Текст статьи должен быть логически законченным и содержать новые научные и практические результаты. От одного автора допускается не более двух статей.

После получения статьи, она отправляется на рецензирование. С результатами рецензирования авторы могут ознакомиться в CMT, при необходимости устранить замечания рецензентов и отправить для повторного рецензирования.

Оргкомитет оставляет за собой право отказать в приеме статьи в случае, если статья не будет соответствовать требованиям оформления и тематике конференции, а также, если будет отсутствовать заявка доклада, соответствующая этой статье.

КОНКУРС ДОКЛАДОВ МОЛОДЫХ УЧЁНЫХ

Соавторами доклада, представляемого на конкурс докладов молодых учёных, могут быть учёные со степенями и званиями, но непосредственно представлять доклад должны авторы в возрасте до 35 лет, не имеющие степеней и званий.

Для того, чтобы принять участие в конкурсе научных докладов молодых учёных, необходимо:

1) заполнить заявку на участие в конференции, в которой чётко указать своё желание принять участие в данном конкурсе;
2) написать статью для публикации в Сборнике научных трудов и загрузить на сайте CMT;
3) заполнить, подписать, отсканировать и отправить по почте письмо о согласии;
4) лично представить доклад на конференции.

КОНКУРС ПРОЕКТОВ МОЛОДЫХ УЧЁНЫХ

Принимать участие в конкурсе проектов молодых учёных могут проекты прикладных интеллектуальных систем и систем, ориентированных на поддержку проектирования интеллектуальных систем, при этом представлять проект на конкурсе должен молодой учёный в возрасте до 35 лет, не имеющий учёной степени.

Для того, чтобы принять участие в конкурсе программных продуктов молодых учёных, необходимо:

1) заполнить заявку на участие в конференции), в которой чётко указать своё желание принять участие в данном конкурсе;
2) написать статью для публикации в Сборнике научных трудов и загрузить на сайте CMT;
3) лично представить доклад на конференции;
4) провести выставочную презентацию, разработанного программного продукта.

КОНКУРС СТУДЕНЧЕСКИХ ПРОЕКТОВ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

В конкурсе студенческих проектов могут принимать участие проекты, разработчиками которых являются студенты и магистранты высших учебных заведений, консультантами и руководителями проекта могут быть лица, имеющие научную степень и звание. Для того, чтобы принять участие в данном конкурсе, необходимо:

1) ознакомиться с положением о конкурсе студенческих проектов (http://conf.ostis.net);
2) заполнить заявку на участие в конкурсе студенческих проектов (http://conf.ostis.net);
3) подготовить описание проекта (http://conf.ostis.net).
4) выслать заявку на участие в конкурсе и описание проекта по электронному адресу конкурса студенческих проектов: ostis.stud@gmail.com.

ПУБЛИКАЦИЯ МАТЕРИАЛОВ КОНФЕРЕНЦИИ

Оргкомитет конференции предполагает публикацию статей, отобранных Программным комитетом по результатам их рецензирования, в Сборнике научных трудов OSTIS в печатном виде и на официальном сайте сборника http://proc.ostis.net в электронном виде**.**

По результатам рецензирования автор отправляет оргкомитету письмо о согласии, которое предусматривает дальнейшую возможность размещения статей, вошедших в сборник научных трудов, в открытом электронном доступе на иных ресурсах по усмотрению редакции сборника.

С 2020 года Сборник научных трудов OSTIS включен в Перечень научных изданий Республики Беларусь для опубликования результатов диссертационных исследований (Перечень ВАК РБ) по технической отрасли наук (информатика, вычислительная техника и управление).

Кроме того, по итогам конференции планируется издание Сборника научных трудов OSTIS в серии «Communications in Computer and Information Science» (CCIS) издательства Springer. Подробная информация об этом приведена на сайте конференции (http://conf.ostis.net).

КЛЮЧЕВЫЕ ДАТЫ КОНФЕРЕНЦИИ

1 марта 2023 г.	начало подачи материалов для участия в конференции
13 марта 2023 г.	срок получения материалов для участия в конференции Оргкомитетом
27 марта 2023 г.	срок предоставления рецензий на статьи
10 апреля 2023 г.	срок принятия решения о публикации присланных материалов и рассылки приглашений для участия в конференции и сообщение о включении статьи в Сборник научных трудов OSTIS
13 апреля 2023 г.	размещение на сайте конференции http://conf.ostis.net проекта Программы конференции OSTIS-2023
17 апреля 2023 г.	размещение на сайте конференции http://proc.ostis.net Сборника научных трудов OSTIS
20 апреля 2023 г.	регистрация участников и открытие конференции OSTIS-2023
20-22апреля 2023 г.	работа конференции OSTIS-2023
2 мая 2023 г.	публикация фоторепортажа и отчёта о проведённой конференции на сайте конференции: http://conf.ostis.net
20 мая 2023 г.	загрузка материалов сборника конференции в РИНЦ

ФОРМИРОВАНИЕ ПРОГРАММЫ КОНФЕРЕНЦИИ

Программа конференции формируется Программным комитетом по результатам рецензирования, представленных статей, а также на основании подтверждения автора(-ов) статьи о прибытии на конференцию.

КОНТАКТНЫЕ ДАННЫЕ ОРГАНИЗАТОРОВ КОНФЕРЕНЦИИ OSTIS

Вся необходимая информация по предстоящей и предыдущих конференциях OSTIS находится на сайте конференции http://conf.ostis.net, а также на сайте материалов конференции http://proc.ostis.net.

По вопросам участия в конференции и решения спорных вопросов обращайтесь: ostisconf@gmail.com.

Методическая и консультативная помощь участникам конференции осуществляется только через электронную почту конференции.

Конференция проходит в Республике Беларусь, г. Минск.

Оргкомитет конференции находится на кафедре интеллектуальных информационных технологий Учреждения образования «Белорусский государственный университет информатики и радиоэлектроники (БГУИР) – г. Минск, ул. Платонова, 39, 5-ый учебный корпус БГУИР.