Versioning Model of Neural Network Problem-Solving Methods in Intelligent Systems

Mikhail Kovalev
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
michail.kovalev7@gmail.com

***Abstract*—In the article, the design process of neural network problem-solving methods in the knowledge bases of intelligent systems is considered. The versioning model of neural network problem-solving methods, described in a specialized language for representation of neural network problem-solving methods, is proposed.**

***Keywords*—problem-solving method, ontological approach, neuro-symbolic AI, artificial neural network**

I. Introduction

The modern development of all directions of *Artificial intelligence* is aimed at building *intelligent systems* that automate more and more complex human activities. The current state in the field of developing *intelligent systems of a new generation*[1] shows that such systems should provide:

- unification of representation and coherence of different knowledge types and problem-solving methods;
- integration and convergence of different problem-solving methods in a single knowledge base to ensure consistency in the semantics of that set of methods;
- representation and interpretation of as many classes of problem-solving methods (programs) as possible.

Integration of different problem-solving methods in a single knowledge base guarantees consistency of semantics of this set of methods. When solving problems using such methods, the system does not communicate with the external environment by transferring input and output data. Instead, a single knowledge base allows the system to track changes in input knowledge in real time using a wide range of methods, which provides the ability to introspect and explain the decisions made by the system. A single *knowledge base* for *problem-solving methods* and *knowledge* used to solve them allows the system to reflect on the process of problem solving, to explain the reasons for its solutions, and to find mistakes there.

The actively developing *class of problem-solving methods* is *artificial neural networks* (ANNs). This is conditioned, on the one hand, by the rapid development of the theoretical foundations of artificial neural networks and on the other hand, by the increasing computing power of the machines used to train them.

Impressive results have been obtained in problem solving with artificial neural networks [2]. Among the positive characteristics of *ANNs* are their ability to effectively solve problems in the absence of known regularities, as well as their ability to solve problems without necessarily developing problem-oriented *problem-solving methods*.

However, there are serious problems with neural network problem-solving methods:

- Heuristic nature of the design process of *neural network problem-solving methods*. The process of selecting *ANNs* architectures and their training parameters places high demands on the knowledge level of *ANNs* engineers.
- Lack of explicit allocation of semantic connections between knowledge in the process of problem solving. They are highlighted implicitly, statistically, based on the data that was used for training. Lack of explicit allocation of meaning leads to the problem of the *"black box"* [3]. An entire field of *Explainable AI* has emerged, in which researchers attempt to explain the *ANNs* solutions [4], [5].

Formalization of *ANNs* in the *knowledge base* of the *intelligent system* together with other *problem-solving methods* allows negating the listed *ANNs* problems, since in such systems, the design problem of *ANNs* and the explanation problem of solutions for these *ANNs* are represented in one form for the whole knowledge base and can be solved using any of the represented *problem-solving method* from this *knowledge base*.

Frequently, the *ANN* is actively changed during the design and interpretation process (configuration of connections, number of layers, synapse weights, activation functions, etc.). To solve the problem of *ANNs* design, the system must be able to analyze the solutions of the same problem on different versions of the same *neural network problem-solving method* in order to evaluate the success of certain solutions in the design of this method, for example, the success of selection of activation functions, training sample, training algorithm, configuration of connections in layers, etc.

The purpose of this article is to develop an approach to *versioning of neural network problem-solving methods* in the *knowledge base* of the *intelligent system*.

II. Proposed approach

In order to solve the above problems, the OSTIS Technology is proposed. 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 [6].

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.

Approaches to ***integration of ANNs with knowledge bases*** in *ostis-systems* are considered in [7]. Input-output integration approaches have been tested and described in [8], [9]. Full *integration of ANNs* with *knowledge bases*, i.e., using neural network problem-solving methods by formalizing them in an ostis-system knowledge base, are described in [10], which describes the *Denotational* and *Operational semantics* of the *Language for representation of neural network problem-solving methods (Neuro-SCP)*. The present work is a development of this language.

The *Language for representation of neural network problem-solving methods* allows representing and interpreting neural network methods in the ostis-system memory. This language is a sublanguage of the *SCP Language* [11]. Any method represented in the *SCP Language* is a fragment of the knowledge base, so the problem of *versioning of neural network problem-solving methods* in the *knowledge base* of the *intelligent system* is reduced to the problem of versioning any *knowledge base fragments*.

During the existence of the *intelligent system*, the state of fragments of its *knowledge base* (that is, the configuration of connections between signs in this knowledge base), as well as the neural network problem-solving method represented in the knowledge base, can change over time [12]. The need to account for the dynamics of knowledge changing over time in *knowledge bases* of *intelligent systems* is conditioned by the qualities inherent in intelligent systems, as well as the range of problems they solve. *Intelligent systems* must:

- maintain the relevance, adequacy, and accuracy of the knowledge stored in it at any point in time;
- remember the history of user and developer actions performed on *fragments* of the knowledge base in order to analyze them and support decision-making in other problems;
- allow performing reverse actions in case of abnormal situations;
- allow verifying the sources of unreliable knowledge and warn about inconsistencies in *knowledge bases*;
- adapt to the characteristics of its users and other *intelligent systems*;
- plan and initiate different kinds of problem solving.

Thus, to provide a higher level of intelligence of the system and support its life cycle, it is necessary to specify a set of methods and tools that allow solving these problems quickly and efficiently. One of such means for solving these problems is a ***Subsystem for versioning of knowledge base fragments***, embedded in any *intelligent system*, developed on the principles of the *OSTIS Technology* (i.e., in ostis-system), providing continuous versioning for various knowledge base fragments and analysis of their states.

A *knowledge base fragment* means a formal specification of any entity or concept sign represented in the *knowledge base* of a given system. That is, a *knowledge base fragment* is nothing but some semantic neighborhood or structure that includes knowledge about an object in the *subject domain of the knowledge base* of this system. The process of *versioning of a knowledge base fragment* implies a complete representation and description of its states, as well as providing capabilities and tools for processing and analyzing the states of this *knowledge base fragment*.

The *state of the knowledge base fragment* means the integration of the results of actions performed on this *knowledge base fragment* since its existence in the *knowledge base* (that is, since the initial state of this knowledge base fragment).

The *versioning of the knowledge base fragment* requires strict identification of all states from the beginning of the existence of this *knowledge base fragment*, that is, it requires the construction of the *bijective correspondence* between the *state of the knowledge base fragment* and its unique identifier in the whole knowledge base of the system [13]. In the process of *versioning of the knowledge base fragment*, a tree-like linear structure of states of this knowledge base fragment (***tree of states of knowledge base fragments***) and a tree-like linear structure of state identifiers of this knowledge base fragment (***tree of state identifiers of knowledge base fragments***) are built. Both structures are represented in the SC-code.

The Subsystem for versioning of knowledge base fragments consists of the following components:

- *specification for the versioning model of knowledge base fragments* (i.e., documentation (knowledge bases) describing methods, tools, and algorithms of *versioning of knowledge base fragments*);
- *problem solver* [14], including:
  - an *Abstract sc-agent for generating the initial state of a knowledge base fragment in its state stack*;
  - an *Abstract sc-agent for integrating changes with the current state of a knowledge base fragment and adding the resulting state of the knowledge base fragment to its state stack*;
  - an *Abstract sc-agent for identifying the state of the knowledge base fragment in the stack of state identifiers of a given knowledge base fragment*;
  - an *Abstract sc-agent for getting the state of a knowledge base fragment by its identifier*;
  - an *Abstract sc-agent for getting the version of a knowledge base fragment by its identifier.*
- *program interface* and *user interface* to use the *versioning of knowledge base fragments.*

The *state of knowledge base fragments* is pairs of two sets. The elements of the first set are the membership arcs of the certain substructures for decomposing the more general structure, which in the version of this *state of the knowledge base fragment* have not been changed. The elements of the second set are the membership arcs of the certain substructures for decomposing the more general structure, which have been changed in this version. Both sets can be represented as a role relation (attribute set), denoting which and in what way the certain structures of the general structure of states are represented. These two sets play an important role in the operational semantics used when implementing agents to work with the history of structure changes.

Reconstructing a version by the *state of the knowledge base fragment* takes only one traversal of the subtree, i.e., a traversal of this state. Structures located in the hierarchy lower than the structure, which includes a membership arc of the composition of a more general structure, which in turn is an element in the attribute set of membership arcs of changed (unchanged) structures, are considered to be changed (unchanged) until an arc from the second attribute set is encountered, and those above the arc are considered to be unchanged (changed) until an arc from the first attribute set is encountered. This approach is easy to implement and does not require a large number of copies for bindings of decomposition and membership arcs for changed structures, compared to the previous cases.

Let us consider an example of versioning a particular neural network method represented in *Neuro-SCP* and described in [10]. This method solves the classical "EXCLUSIVE OR" problem [15].

In *Neuro-SCP*, the implementation of this *ANN* is reduced to a single layer interpretation operator, for which the *matrix of synapse weights*, *threshold*, and *activation function* are defined. In Figure 1, the *neuro-SCP* interpretation operator of the *layer of a single-layer perseptron*, solving the "EXCLUSIVE OR" problem, is shown.

In Figure 2, the *single-layer perseptron* scheme, which solves this problem and from which the *Neuro-SCP* operator has been described, is demonstrated.

Any of these elements can be changed in the process of training and reconfiguration of the *ANN*. Within the proposed approach, such *ANN* will correspond to its own *tree of states*. Let us suppose that a given *ANN* has been trained and its *synaptic weight matrix* of a single *layer* has changed.

In Figure 3, a *tree of states* that stores two versions of the same *ANN* — before training and after training — is shown. The root of this tree is an instance of the history class. There are two states associated with the root, each of which is an oriented pair of sets.

For each state, the time of the beginning of the state existence is set. All elements describing the states are *temporal*, since they exist temporarily and each state can be replaced by another.

For the first state, the first set was empty, since it is the first state in the tree and it cannot refer to previous states. The second set considered above describes all elements that were added in this version. Since it is the first state, this set describes all membership arcs involved in the description of the operator.

The first set of the second state — the set of elements that have not changed from the previous state — is described using the constructions involved in describing the second set of the first state. The second set of the second state, on the other hand, describes only the changes or, to be more precise, only the membership arcs that have been added in the new state. Thus, it is possible to understand that the weight matrix has changed in the new state. It can be assumed that the *ANN* has been retrained.

The use of ANNs as a problem-solving method implies the use of an already designed and trained ANNs. However, the presence of a neural network method description language in ostis-system memory opens the way for automation of the processes of designing and training ANNs. Such automation is represented by separate classes of problems and the corresponding skills for their solution.

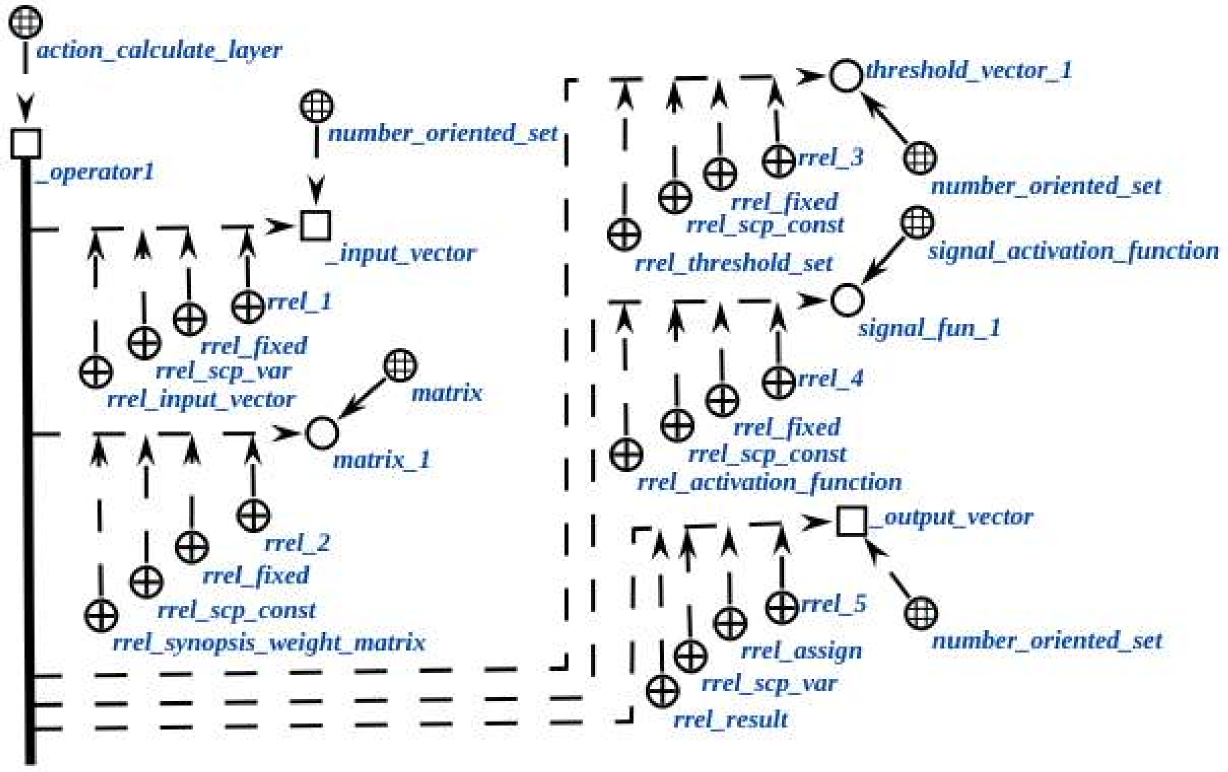

Figure 1. Operator for interpreting the ANN layer on Neuro-SCP

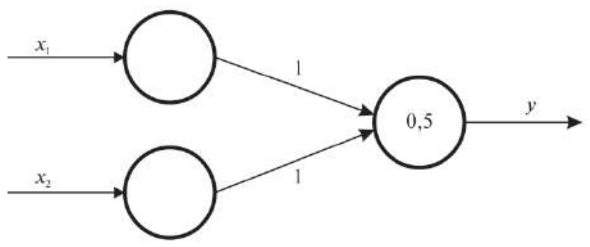

Figure 2. Scheme of a single-layer perseptron solving the “EXCLUSIVE OR” problem [15]

III. Conclusion

The described versioning model allows storing and restoring any state not only of the *neural network method* described with *Neuro-SCP* in the *knowledge base* but also any *knowledge base fragment* in principle with minimal effort.

The model can be used to describe not only what has been changed but also at what time and under what influences. If necessary, each state can be described from various angles — efficiency, validity, operational history, etc.

In the case of ***convergence and integration of ANNs with knowledge bases of intelligent systems***, such a model will allow considering *ANNs* not only as a problem-solving method but also as a design object. The automation of such design will become a task that can be posed by the same intelligent system, in which the neural network is described and interpreted.

The availability of unification of the knowledge representation describing the problem and the methods of solving these problems allows the whole arsenal of *problem-solving methods of the intelligent system* to be used for design. Such methods can achieve good results in design automation by accessing the knowledge describing the problem to be solved by the designed *ANNs*, the context of the problem to be solved, the history of the intelligent system solving similar problems, etc.

A further development of this work is the design and implementation of *intelligent design framework of ANNs*, which will automate various activities for ANNs designers.

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 Golenkov, Vladimir Golovko, Aliaksandr Kroshchanka, and Nikita Zotov.

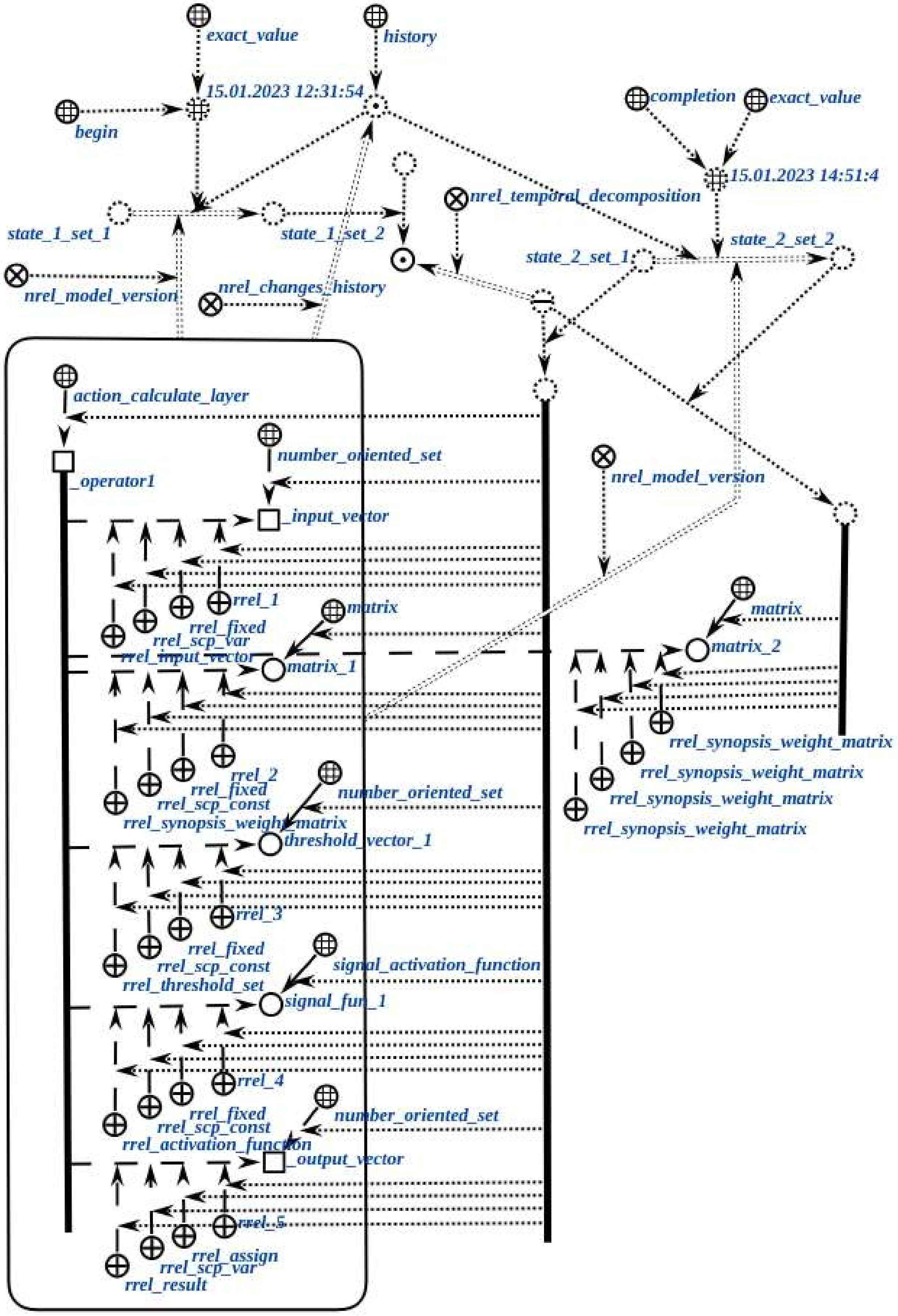

Figure 3. Example of an ANN tree of states

References

[1] V. V. Golenkov and N. A. Gulyakina, “Next-generation intelligent computer systems and technology of complex support of their life cycle,” *Open semantic technologies for intelligent systems*, pp. 27–40, 2022.

[2] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Q. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, “Review of deep learning: concepts, cnn architectures, challenges, applications, future directions,” *Journal of Big Data*, vol. 8, 2021.

[3] D. Castelvecchi, “Can we open the black box of AI?” *Nature News*, vol. 538, no. 7623, Oct 2016.

[4] 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

[5] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[6] D. Shunkevich, “Ontology-based design of hybrid problem solvers,” *Open Semantic Technologies for Intelligent Systems*, pp. 101–131, 2022.

[7] V. A. Golovko, V. V. Golenkov, V. P. Ivashenko, V. V. Taberko, D. S. Ivaniuk, A. A. Kroshchanka, and M. V. Kovalev, “Integration of artificial neural networks and knowledge bases,” *Open semantic technologies for intelligent systems*, pp. 133–145, 2018.

[8] V. Golovko, A. Kroshchanka, M. Kovalev, V. Taberko, and D. Ivaniuk, “Neuro-symbolic artificial intelligence: Application for control the quality of product labeling,” *Open Semantic Technologies for Intelligent System*, pp. 81–101, 10 2020.

[9] A. Kroshchanka, V. Golovko, E. Mikhno, M. Kovalev, V. Zahariev, and A. Zagorskij, “A Neural-Symbolic Approach to Computer Vision,” *Open Semantic Technologies for Intelligent Systems*, pp. 282–309, 2022.

[10] M. Kovalev, “Convergence and integration of artificial neural networks with knowledge bases in next-generation intelligent computer systems,” *Open semantic technologies for intelligent systems*, pp. 173–186, 2022.

[11] V. Golenkov, N. Gulyakina, and D. Shunkevich, *Otkrytaya tekhnologiya ontologicheskogo proektirovaniya, proizvodstva i ekspluatatsii semanticheski sovmestimykh gibridnykh intellektual’nykh komp’yuternykh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems]*, V. Golenkov, Ed. Minsk: Bestprint, 2021.

[12] V. Ivashenko, “Semanticheskoe protokolirovanie processov obrabotki znanij [Semantic logging of knowledge processing processes],” in *Information Technologies and Systems 2017 (ITS 2017) : Republic of Belarus, Minsk, 25 october 2017 year)*, L. Y. Shilin, Ed. Minsk: BSUIR, 2017. [Online]. Available: https://libeldoc.bsuir.by/handle/123456789/27633

[13] ——, “Identifikaciya binarno porozhdaemyh sobytij processov obrabotki znanij dlya semanticheskogo protokolirovaniya [Identification of binary generated events of knowledge processing processes for semantic logging],” in *Information Technologies and Systems 2018 (ITS 2018) : Republic of Belarus, Minsk, 25 october 2018 year)*, L. Y. Shilin, Ed. Minsk: BSUIR, 2017. [Online]. Available: https://libeldoc.bsuir.by/handle/123456789/33352

[14] D. Shunkevich, “Agentno-orientirovannye reshateli zadach intellektual’nyh sistem [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]*, pp. 119–132, 2018.

[15] V. A. Golovko and V. V. Krasnoproshin, *Nejrosetevye tekhnologii obrabotki dannyh [Neural network data processing technologies]*. Minsk : Publishing House of the BSU, 2017, (In Russ.).

Модель версионирования нейросетевых методов решения задач в интеллектуальных системах

Ковалёв М. В.

В статье рассматривается процесс проектирования нейросетевых методов решения задач в базах знаний интеллектуальных систем. Предложена модель версионирования нейросетевых методов решения задач, описанных на специализированном языке представления нейросетевых методов решения задач.

Received 29.03.2023