Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра интеллектуальных информационных технологий

С. А. Самодумкин, М. Д. Степанова, Д. Г. Колб

ПРИКЛАДНЫЕ ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Рекомендовано УМО по образованию в области
информатики и радиоэлектроники
в качестве учебно-методического пособия
для специальности 1-40 03 01 «Искусственный интеллект»

Минск БГУИР 2014

УДК 004.896(076)
ББК 32.813я73
С17

Р е ц е н з е н т ы:
кафедра интеллектуальных систем
Белорусского государственного университета
(протокол №5 от 21.11.2013);

ведущий научный сотрудник государственного научного учреждения
«Объединенный институт проблем информатики
Национальной академии наук Беларуси»,
кандидат физико-математических наук, доцент Ю. В. Поттосин

Самодумкин, С. А.

С17 Прикладные интеллектуальные системы : учеб.-метод. пособие / С. А. Самодумкин, М. Д. Степанова, Д. Г. Колб. – Минск : БГУИР, 2014. – 103 с. : ил.

ISBN 978-985-488-975-7.

Учебно-методическое пособие содержит материалы по принципам и методам разработки прикладных интеллектуальных систем. Рассмотрены наиболее популярные подходы к организации интеллектуальных систем. Отдельный раздел посвящен разработке интеллектуальных систем для сети Интернет.

Предназначено для студентов специальности «Искусственный интеллект» всех форм обучения.

УДК 004.896(076)
ББК 32.813я73

ISBN 978-985-488-975-7

Содержание

СПИСОК ОСНОВНЫХ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

АГ – автоматическое гипотезирование
АДТ – автоматическое доказательство теорем
БГШ – базовый графовый шаблон
БЗ – база знаний
ГИС – географическая информационная система, геоинформационная система
ИАД – интеллектуальный анализ данных
ИИ – искусственный интеллект
ИС – интеллектуальная система
ИСР – информационные системы руководства
ЛПР – лицо, принимающее решения
МАС – многоагентные системы
ПИС – прикладные интеллектуальные системы
ППП – пакеты прикладных программ
ПР – принятие решения
ПС – продукционная система
РАД – разведочный анализ данных
РП – решающее правило
СОЗ – системы, основанные на знаниях
СПР – системы принятия решений
ЭС – экспертная система

Предисловие

Данное учебно-методическое пособие – одно из серии пособий, разрабатываемых на кафедре интеллектуальных информационных технологий Белорусского государственного университета информатики и радиоэлектроники. Курс «Прикладные интеллектуальные системы» изучают студенты специальности 1-40 03 01 «Искусственный интеллект» всех форм обучения.

Сложные системы, управление сложными системами, модели – эти понятия в настоящее время встречаются и используются практически в любой области деятельности человека. Данное пособие должно помочь студенту в ряде возникающих проблем и вопросов, таких, как:

– необходимость применения методов искусственного интеллекта при решении прикладных задач;

– представление данных и знаний о системе в некоторых формальных конструкциях, близких к языку предметной области;

– перевод исходной системы и решаемых задач на язык описания модели,

– принципы построения систем, использующих знания для анализа и решения слабо структурированных (неформализованных) задач.

Основные задачи данного пособия можно сформулировать следующим образом:

– ознакомить с типологией задач интеллектуализации прикладных систем;

– ознакомить с интеллектуальными компонентами систем, их структурой и назначением;

– ознакомить со сферами применения прикладных интеллектуальных систем;

– дать представление о методах описания предметной области на естественном и модельных языках;

– обучить методам представления знаний и вывода на знаниях в интеллектуальных системах.

Область знаний об интеллектуальных системах в настоящее время не ограничивается разделами данного пособия. Некоторые темы, логически связанные с излагаемым материалом, не включены в пособие, т. к. входят в учебные программы других дисциплин: «Проектирование баз знаний», «Логические основы интеллектуальных систем», «Модели решения задач в интеллектуальных

системах», «Технология проектирования интеллектуальных систем», «Интеллектуальные Интернет-технологии».

Последовательность изложения материала в настоящем пособии выглядит следующим образом:

1. Общие сведения об интеллектуальных системах (структура и основные компоненты прикладных интеллектуальных систем (ПИС), типология ИС).

2. Системы, основанные на знаниях (моделирование знаний предметной области как на основе моделей знаний, так и с помощью онтологий, способы построения онтологий, примеры ПИС, semantic web).

3. Методы принятия решений в ИС.

Настоящее учебно-методическое пособие разработано для студентов, обучающихся по специальности «Искусственный интеллект». Этот материал рекомендуется студентам, магистрантам инженерно-технических специальностей, изучающим информационные технологии и методы их интеллектуализации.

1. Прикладные интеллектуальные системы (ПИС) и их назначение

1.1. Интеллектуальные системы (ИС) как класс прикладных систем

Программная система, как правило, представляется коллекцией программ, предназначенных для решения задач определенного типа и объединяемых общим интерфейсом. Адаптация того или иного типа программных систем к некоторой прикладной области приводит к получению прикладной системы, имеющей своих пользователей, конкретное назначение и применение. Основными требованиями, предъявляемыми к прикладным системам, являются: необходимость учета проблемной специфики и проблемных знаний при разработке системы, обеспечение соответствия уровня функционирования системы профессиональному уровню проблемной области и постоянная коррекция уровня функционирования на стадии сопровождения.

Интеллектуальные системы являются одним из классов прикладных систем. ИС называются обычно такие технические или программные системы, которые обладают способностью решать задачи, традиционно считающиеся творческими, интеллектуальными, принадлежащие конкретной предметной области. Знания о предметной области хранятся в памяти интеллектуальной системы. Из функций для систем искусственного интеллекта (ИИ) наиболее характерны экспертность, структуризация, хранение и поиск информации, управление исполнением.

К системам ИИ относятся: системы обработки текстов естественного языка, системы речевого общения, системы обработки визуальной информации, информационные и поисковые системы, системы машинного перевода, системы принятия и планирования решений, экспертные системы, оболочки экспертных систем и др.

Применение методов и средств искусственного интеллекта при создании конкретных прикладных систем в настоящее время является одной из актуальных задач современной информатики, т. к. указанные методы позволяют перейти к принципиально более качественному и эффективному классу интеллектуальных систем. В особенности это касается систем, обрабатывающих большие объемы сложно структурированной информации и требующих применения

различных методов решения неформализованных задач. Развитие информационных технологий дает новые средства для разработки систем принятия решений. Методы и системы принятия решений (СПР) предназначены для поиска лучших вариантов решений.

1.2. Принципы организации ИС

Рассмотрим класс прикладных систем, которые соответствуют следующим пяти принципам организации управляющей структуры [1].

Первый принцип. *Наличие тесного информационного взаимодействия систем с внешним миром с использованием специально организованных информационных каналов связи*.

Этот принцип подчеркивает непосредственную связь интеллектуальных систем с внешним миром. Находясь в непрерывном взаимодействии с внешним миром, интеллектуальные системы получают из него всю необходимую информацию для принятия решений и пополнения знаний. Сама система, в свою очередь, может оказывать на внешний мир непосредственное активное воздействие в результате реализации собственного поведения. Модель знаний о внешнем мире должна предполагать возможность изменений внешнего мира и знаний о нем в результате собственных на него воздействий. Выполнение принципа информационного взаимодействия системы с внешним миром означает, что модели типа моделей состояний, вероятностных описаний, игр автоматов со средой и т. п. для представления событий реального мира недостаточно пригодны. Именно в этом и состоит принципиальное отличие интеллектуальных управляющих систем.

Если в традиционных системах внешние незапланированные, неизвестные воздействия являются скорее негативным фактором, то в ИС внешние воздействия могут иметь также характер информационного ресурса. Этот ресурс может быть использован системой для самоорганизации. Находясь в тесном взаимодействии с изменяющимся внешним миром, система получает информацию для пополнения знаний и принятия решений.

Второй принцип. *Принципиальная открытость систем с целью повышения интеллектуальности и совершенствования собственного поведения системы в динамически меняющемся мире.*

Принципиальная открытость систем в соответствии со вторым принципом (с целью повышения интеллектуальности и совершенствования собственного поведения) обеспечивается наличием таких уровней высшего ранга в иерархической структуре, как самонастройка, самоорганизация и самообучение. Система знаний интеллектуальной системы состоит из двух частей: 1) постоянных (проверенных) знаний, которыми система обладает и постоянно пользуется; 2) временных (проверяемых) знаний, с которыми она экспериментирует в процессе обучения. Знания второго типа либо отбрасываются системой, либо переходят в знания первого типа в зависимости от результатов анализа своего поведения во внешнем мире. Знания первого типа хранятся в базе знаний. Выполнение второго принципа требует организации в управляющей системе процесса приобретения и пополнения знаний.

Третий принцип. *Наличие механизмов прогноза изменений внешнего мира и собственного поведения системы в динамически меняющемся внешнем мире.*

В соответствии с третьим принципом управления систему нельзя считать в достаточной мере интеллектуальной, если она не обладает возможностью прогноза изменений самого внешнего мира и собственного в нем поведения (в динамически меняющемся внешнем мире). Система без прогноза (функционирующая именно в динамически меняющемся внешнем мире) может попасть в критическую ситуацию, из которой не сможет найти выхода. Это происходит из-за возникновения временных ограничений на работу механизмов формирования управляющих воздействий, определяющих ее поведение, адекватное сложившейся ситуации. Примером могут служить автономно функционирующие интеллектуальные робототехнические системы в условиях экстремальных ситуаций.

Четвертый принцип. *Наличие у системы многоуровневой иерархической структуры, построенной в соответствии с правилом: повышение интеллектуальности системы и снижение требований к ее точности по мере повышения ранга иерархии и наоборот.*

Четвертый принцип дает возможность указать пути построения моделей сложных систем в тех случаях, когда неточность знаний о модели управляемого объекта или модели его поведения можно скомпенсировать за счет увеличения числа уровней интеллектуальности, а также за счет использования механизмов

принятия решений в условиях неопределенности в соответствующих алгоритмах управления проектируемых систем.

Более высокого ранга уровень решает исходную, более сложную, более творческую задачу, которая зачастую и не может быть решена точно, чисто алгоритмически. Это происходит из-за отсутствия точной подстановки или подходящего алгоритма. Решением в данном случае является взаимосвязанная совокупность более простых подзадач. Неточность или неполнота знания о модели объекта, самой системе компенсируется введением дополнительных уровней интеллектуальности.

Пятый принцип. *Сохраняемость функционирования (возможно, с некоторой потерей качества или эффективности) при разрыве связей или потере управляющих воздействий от высших уровней иерархии управляющей структуры*.

Пятый принцип устанавливает лишь частичную потерю работоспособности, но не прекращение функционирования при отказах в работе высших уровней иерархии системы. Сохранение автономного функционирования в рамках более простого поведения системы, характерного для нижних уровней структуры управления, чрезвычайно важно для автономно функционирующих систем в реальном внешнем мире. Примером могут служить те же интеллектуальные роботы.

1.3. Определение интеллектуальной системы

На основании приведенных пяти принципов организации структуры можно дать определение интеллектуальной системы.

Определение 1. *Системы, организованные и функционирующие в соответствии со сформулированными пятью принципами (в полном их объеме), называются системами, обладающими свойством «интеллектуальности в большом»*.

Из определения 1 следует, что системы, обладающие свойством «интеллектуальности в большом», должны иметь многоуровневую иерархическую структуру со следующими уровнями иерархии (в порядке понижения ранга):

– уровень самоорганизации (самоперестройки по результатам обучения);

– уровень обучения;

– уровень прогноза событий;

– уровень адаптации;

– уровень формирования решений;

– уровень работы с базами событий и знаний;

– уровень планирования операции по реализации сформированного решения;

– исполнительный уровень.

Каждый из перечисленных уровней в реальной системе может состоять из нескольких подуровней. При этом на самом нижнем уровне обычно используются традиционные модели систем управления. Все остальные уровни более высокого ранга можно рассматривать как надстройку над традиционными управленческими моделями, отвечающую требованиям современной информационной технологии работы со знаниями. Минимальная надстройка может содержать только базу знаний, состоящую из нескольких продукционных правил. В простейшем случае могут отсутствовать уровни принятия решений и планирования (или эти функции могут быть совмещены с функциями обработки правил).

Определение 2. С*истемы, структурно не организованные в соответствии с приведенными выше пятью принципами, но использующие при функционировании знания как средство преодоления неопределенности входной информации, модели объекта или его поведения, называются системами, обладающими свойством «интеллектуальности в малом»*.

Определение 2 соответствует общепринятому определению интеллектуальной системы как системы, ориентированной на обработку знаний с целью поиска решения задач. Определения 1 и 2 устанавливают границы уровней интеллектуальности управляющих систем. Степень интеллектуальности управляющих систем внутри этих границ можно определять по наличию или отсутствию тех или иных уровней, введенных выше.

В системах, интеллектуальных в большом, интеллектуальность проявляется в таких направлениях, как управление неопределенностью, обучение и адаптация. Это сложные системы с многоуровневой иерархической структурой, способные к формированию адекватных сложившейся ситуации решений в условиях неопределенности используемой информации.

На основании кибернетического подхода можно дать следующее определение интеллектуальной системы. *Под интеллектуальной системой понимает-*

ся объединенная информационным процессом совокупность технических средств и программного обеспечения, работающая во взаимодействии с человеком (коллективом людей) или автономно, способная на основании сведений об окружающей среде и собственном состоянии при наличии знаний и мотивации синтезировать цель, принимать решение о действии и находить рациональные способы достижения цели.

1.4. Классификация прикладных интеллектуальных систем

Классификация прикладных интеллектуальных систем может быть произведена по различным принципам [2, 3, 4]. Рассмотрим классификацию ПИС на основании *решаемых задач*.

Интерпретация данных. Это одна из традиционных задач для экспертных систем. Под интерпретацией понимается процесс определения смысла данных, результаты которого должны быть согласованными и корректными. Обычно предусматривается многовариантный анализ данных.

Диагностика. Под диагностикой понимается процесс соотношения объекта с некоторым классом объектов и/или обнаружение неисправности в некоторой системе.

Мониторинг. Основная задача мониторинга – непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы.

Проектирование. Проектирование состоит в подготовке спецификаций на создание «объектов» с заранее определенными свойствами. Под спецификацией понимается весь набор необходимых документов – проект, чертеж, пояснительная записка и т. д. Основная проблема здесь – это получение четкого структурного описания знаний об объекте. Для организации эффективного проектирования необходимо формировать не только сами проектные решения, но и мотивы их принятия. Таким образом, в задачах проектирования тесно связываются два основных процесса, выполняемых в рамках соответствующей ЭС: процесс вывода решения и процесс объяснения.

Прогнозирование. Прогнозирование позволяет предсказывать последствия некоторых событий или явлений на основании анализа имеющихся данных. Прогнозирующие системы логически выводят вероятные следствия из заданных ситуаций. В прогнозирующей системе обычно используется параметриче-

ская динамическая модель, в которой значения параметров «подгоняются» под заданную ситуацию. Выводимые из этой модели следствия составляют основу для прогнозов с вероятностными оценками.

Принятие решений и планирование. Под *принятием решений* понимается процесс человеческой деятельности, направленный на выбор наилучшего варианта действий. Под планированием понимается нахождение планов действий, соответствующих каждому из возможных вариантов. В таких системах используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности.

Обучение. Под обучением понимается использование компьютера для обучения. Системы обучения служат средством освоения учебного материала, осуществляют контроль знаний при изучении какой-либо дисциплины и подсказывают правильные решения. Они аккумулируют знания о гипотетическом «ученике» и его характерных ошибках, затем в работе они способны диагностировать слабости в познаниях обучаемых и находить соответствующие средства для их ликвидации. Кроме того, они планируют акт общения с учеником в зависимости от успехов ученика с целью передачи знаний.

Управление. Под управлением понимается функция организованной системы, поддерживающая определенный режим деятельности. Такого рода системы осуществляют управление поведением сложных систем в соответствии с заданными спецификациями.

Поддержка принятия решений. Поддержка принятия решения – это совокупность процедур, обеспечивающая лицо, принимающее решения, необходимой информацией и рекомендациями, облегчающими процесс принятия решения. Такие процедуры помогают специалистам выбирать и/или формировать нужную альтернативу среди множества вариантов при принятии решений.

В общем случае все прикладные системы, основанные на знаниях, можно подразделить на системы, решающие задачи анализа, и на системы, решающие задачи синтеза. Задачами анализа являются: интерпретация данных, диагностика, поддержка принятия решения. К задачам синтеза относятся проектирование, планирование, управление. Комбинированными можно считать ИС для обучения, мониторинга, прогнозирования.

Известна также классификация ИС по связи с реальным временем:

– *статические ИС* разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени;

– *квазидинамические ИС* интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени;

– *динамические ИС* работают в сопряжении с датчиками объектов в режиме реального времени с непрерывной интерпретацией поступающих в систему данных.

Существует также классификация по степени интеграции с другими программами:

– *автономные ИС* работают непосредственно в режиме консультаций с пользователем для специфических «экспертных» задач, для решения которых не требуется привлекать традиционные методы обработки данных (расчеты, моделирование и т. д.);

– *гибридные ИС* представляют программный комплекс, агрегирующий стандартные пакеты прикладных программ (например, математическую статистику, линейное программирование или системы управления базами данных) и средства манипулирования знаниями. Это может быть интеллектуальная надстройка над ППП (пакетами прикладных программ) или интегрированная среда для решения сложной задачи с элементами экспертных знаний.

2. Основные направления интеллектуализации прикладных систем и систем принятия решений (СПР)

2.1. Методы искусственного интеллекта в прикладных системах и системах принятия решений

До недавнего времени постановка и решение задач в прикладных системах опирались на традиционные математические модели и методы. Но в задачах управления сложными объектами, планирования, принятия решений, прогнозирования, диагностики зависимости настолько сложны, что не допускают своего обычного аналитического представления или из-за неполноты информации некоторые элементы модели остаются неизвестными.

Это объясняется следующими причинами:

1. Не все цели управления объектом могут быть выражены в виде количественных соотношений.
2. Между рядом параметров, оказывающих влияние на процесс принятия решений, не удается установить точных количественных зависимостей.
3. Процесс принятия решений является многошаговым, и содержание каждого шага не может быть заранее однозначно определено.
4. Существующие способы описания объектов и протекающих в них процессов приводят к столь громоздким конструкциям, что их практическое использование невозможно.

В этих случаях средства ИИ позволяют более успешно справиться с неполнотой информации и отсутствием определенности.

Такие объекты называют по-разному: плохо определенные или слабо структурированные, организационные. Примерами таких объектов управления являются, например, экономические и социальные объекты. Но независимо от названия эти новые объекты обладают рядом присущих только им свойств. Перечислим основные их свойства [5].

Уникальность. Каждый объект обладает такой структурой и функционирует так, что система управления им должна строиться с учетом всех его качеств и к нему нельзя применить какую-либо стандартную процедуру управления. Это обстоятельство резко удорожает построение системы управления, т. к. фактически нужно создавать столько систем управления, сколько объектов мы хотим автоматизировать.

Отсутствие формализуемой цели существования. Не для всех объектов (даже созданных человеком) можно четко сформулировать цель их существования. При управлении городами, отраслями народного хозяйства, регионами, экосистемами возникают затруднения при попытке четко сформулировать цель существования этих объектов. Это приводит к большим сложностям в формировании критерия управления, т. к. в традиционных системах управления он был тесно связан с целью существования объекта. Критерий управления самолетом был основан на достижении им своей цели существования – перевозке грузов и людей по воздуху; управления производством телевизоров – на повышении качества выпускаемой продукции. Поэтому в различных системах управления, создаваемых для объектов нового класса, очень часто можно наблюдать реализацию различных критериев управления.

Отсутствие оптимальности. Следствием того, о чем говорилось в предшествующих двух пунктах, является неправомочность постановки для объектов новой природы классической задачи оптимизации. Из-за отсутствия цели существования (в рамках теории управления) для рассматриваемых объектов нельзя построить объективный критерий управления. Он становится субъективным, зависящим от лица, принимающего решения.

При решении о выборе тех или иных характеристик будущего изделия, при принятии решения о структуре и способах функционирования проектируемой системы невозможно оценить правильность выбора. Отсюда следует, что в этих случаях невозможно говорить об оптимальности получаемых решений. Качество создаваемой системы для управления объектами новой природы может оцениваться только субъективно самим лицом принимающим решение (ЛПР) или их коллективом. Поэтому здесь уместнее говорить о целесообразности результата управления, а не его оптимальности.

Динамичность. Существует обширный класс объектов управления, которые с течением времени изменяют свою структуру и функционирование, т. е. объекты эволюционируют во времени. Эта динамичность должна быть учтена в системах управления подобными объектами в виде эволюции процесса управления.

Неполнота описания. Первая причина неполноты описания объекта состоит в том, что эксперты не могут оценить тот уровень полноты описания, который нужен специалисту по управлению. Как правило, эксперты, знающие

объект управления, не в состоянии сразу же обеспечить информацию, которой бы хватило для создания системы управления объекта.

Вторая причина неполноты описания объекта – незнание некоторых сторон функционирования самими специалистами. Могут возникнуть ситуации, никогда не встречавшиеся ранее. К ним относятся всевозможные аварийные ситуации.

Третья причина неполноты описания объекта – отсутствие четкого понимания функционирования объекта у специалиста-технолога. Выдавая управленцу большое количество информации, технолог тем не менее не сообщает ему самой главной, по которой сам принимает решение.

Четвертая причина неполноты описания объекта – многие особенности функционирования объекта, а иногда и его структуры, не могут быть описаны количественно. Они допускают лишь качественное словесное описание. Переход от качественных описаний к некоторым формальным представлениям должен производиться управленцем, который не всегда в состоянии решить такую сложную проблему.

Активная природа элементов управляемого объекта. Во многих объектах управления люди являются элементами их структуры. Это так называемые организационные системы. В отличие от всех других элементов, образующих объекты, люди функционируют в нем с учетом своих личных интересов и целей, которые могут противоречить целям управления и оказывать обратное воздействие на саму систему управления. Индивидуальное поведение таких элементов структуры практически невозможно учесть при создании системы управления, и требуются специальные приемы для учета их воздействия на функционирование объекта управления.

Использование интеллектуальных компонентов приводит к новым технологиям в принятии решений.

2.2. Типология задач интеллектуализации систем

Одним из направлений повышения эффективности использования программных систем является их интеллектуализация. Интеллектуализация прикладных систем предполагает:

- возможность общения конечного пользователя с системой на формализованном языке предметной области, включая общение аудио- и видеосредствами;
- использование алгоритмов принятия решений на основе исследования проблемы с объяснением этих решений по заданной пользователем неполной или неточной информации;
- возможность манипулирования знаниями;
- использование алгоритмов анализа полученных решений;
- использование поисковых процедур и методов решения задач перебора, связанных с поиском и просмотром большого числа вариантов;
- возможность динамической пространственной визуализации для представления результатов.

Реализация возможностей интеллектуальных СПР может быть осуществлена с помощью:

- систем интеллектуального интерфейса;
- систем управления базами знаний;
- экспертных систем в различных областях деятельности людей;
- систем поддержки принятия решений;
- мультимедийных баз знаний и данных по областям применения;
- средств интеллектуального программирования (языки логического программирования и представления знаний в компьютерах, объектно-ориентированные языки программирования, инструментальные средства разработки программ, системы когнитивной графики); информационно-обучающих систем (интеллектуальные тренажеры, системы деловых игр, консультационные системы).

2.3. Интеллектуальные информационные технологии в прикладных системах и системах принятия решений

Разработка интеллектуальных компонент прикладных систем ведется с использованием интеллектуальных информационных технологий [1, 2, 3]. Изначально под интеллектуальной информационной технологией подразумевались средства обработки информации, не связанные с алгоритмическими вычислениями. Сегодня благодаря многочисленным публикациям в области искусственного интеллекта термин «интеллектуальная информационная техноло-

гия» трактуется одинаково. Достижения в области информационных технологий и, в частности, в искусственном интеллекте вызвали появление новых средств принятия решений, управления и оптимизации сложных систем на основании принимаемых решений. Современная интеллектуальная информационная технология включает следующие основные направления [2]:

1) нейросетевые алгоритмы обработки информации и нейрокомпьютеры;

2) эволюционные (генетические) алгоритмы;

3) системы, реализующие методы, основанные на знаниях;

4) объектно-ориентированные интеллектуальные системы;

5) обработка нечеткой информации и нечеткий вывод;

6) многоагентные системы как средство распараллеливания поиска и обработки информации в интеллектуальных системах;

7) интеллектуальный анализ данных.

В последние годы к перечисленным направлениям стали относить интеллектуальный анализ данных.

Кратко охарактеризуем каждое из этих направлений.

1. Искусственные **нейронные сети** применяются для решения целого класса задач, где используются не уравнения, описывающие функционирование объекта, и не правила, как в традиционных экспертных системах, а опыт. Нейронные сети обладают способностью к обобщению информации, к обучению и самообучению на основе собственного опыта функционирования и т. п. В интеллектуальных компонентах сложных систем применяются несколько типов искусственных нейронных сетей: многослойный перцептрон, сеть Кохонена, сеть Хопфилда и др. Эти модели используются для автоматизации принятия решений в следующих ситуациях:

– когда невозможно построить алгоритм или логическое исчисление из-за сложности учета всех сочетаний факторов;

– когда сложно формализовать закономерности, связывающие условия задачи с результатом.

В мощных интеллектуальных системах могут совмещаться нейросетевые и логические механизмы принятия решений.

В качестве области практического приложения нейросетей в интеллектуальных компонентах прикладных систем можно указать системы искусственного зрения и обработки изображений, системы аэрокосмической съемки, под-

системы автоматизации планирования действий в мониторинговых системах и системах предупреждения чрезвычайных ситуаций.

2. Интеллектуальные компоненты прикладных систем могут быть реализованы и на базе **эволюционных (генетических) алгоритмов,** которые используются для поиска рациональных решений.

Эволюционные алгоритмы основаны на дарвиновских принципах генерации, тестирования и отбора перспективных популяций. Эволюционные алгоритмы оперируют с популяцией индивидов $P(t) = \{ x_1^t, \ldots, x_n^t \}$, где t – номер итерации. Каждый индивид представляет собой некоторое возможное решение из множества допустимых решений S. Каждое решение x_i^t оценивается некоторой мерой его пригодности. На итерации $t + 1$ формируется новая популяция путем отбора более пригодных индивидов (шаг селекции). Некоторые члены этой новой популяции подвергаются преобразованиям (шаг изменений) с помощью генетических операторов. Это делается для того, чтобы образовать новые решения. Преобразования бывают двух типов. Первый тип – это унарные (одноместные) преобразования $m_k : s \to S$ (типа мутаций), которые приводят к появлению новых индивидов путем малых изменений одного индивида. Второй тип преобразований – это преобразования перекрестного типа $c_j : S^n \to S$, которые порождают новые индивиды путем комбинирования составных частей – генов нескольких индивидов. Через несколько поколений могут возникнуть решения, близкие к оптимальным. Генетические алгоритмы используют параллельную обработку множества альтернативных решений и организуют поиск наиболее перспективных из них.

В качестве примера применения эволюционного алгоритма можно привести задачу поиска графа, удовлетворяющего некоторым требованиям (оптимальный маршрут, оптимальная топология коммуникационной сети). Такая задача возникает при реализации механизма вывода в семантических сетях. Граф рассматривается как индивид эволюционного алгоритма. Начальными данными эволюционной программы служит начальная популяция графов, порождаемая случайным образом или эвристически. Задается оценочная функция, которая выражает пригодность каждого графа и формализует отношение предпочтения (лучше, хуже) на множестве индивидов. Мутационные операторы осуществляют преобразования графа. Перекрестные операторы комбинируют структуры

двух или более графов. Если искомый граф должен быть связным и ациклическим (т. е. деревом), некоторый мутационный оператор может удалять ребра и для связывания двух возникших новых подграфов всякий раз добавлять некоторое новое ребро. С помощью оценочной функции устанавливается, не нарушено ли в результате мутаций свойство графа быть деревом. Если порожденный граф этому свойству не удовлетворяет, на шаге селекции необходимо отбросить графы-поддеревья.

Эволюционные алгоритмы применяются в интеллектуальных компонентах ГИС для решения задачи планирования маршрутов перевозок, обеспечивающих рациональное расходование ресурсов.

3. **Системы, основанные на знаниях.** Системы, основанные на знаниях (СОЗ), оперируют моделями, основанными на знаниях экспертов. Вместе с тем СОЗ могут использовать и традиционные алгоритмы, базирующиеся на традиционных математических моделях. Основным механизмом интеллектуализации в системах, основанных на знаниях, является тот или иной механизм рассуждений. В зависимости от этого можно выделить следующие типы систем:

– системы, основанные на правилах;

– системы, в основе которых лежат модели в виде семантических сетей различного рода и фреймов;

– системы, основанные на автоматическом доказательстве теорем;

– системы, основанные на автоматическом порождении гипотез;

– системы, основанные на рассуждениях по аналогии;

– системы, реализующие объектно-ориентированный вывод.

Знания в СОЗ представляются с помощью логических языков или языков представления знаний. Обработка знаний осуществляется специальными инструментальными средствами. В результате обработки получают предпочтительное решение, выбранное из множества допустимых решений.

4. **Объектно-ориентированные интеллектуальные системы** [2, 6] используют декларативно-процедурные формы представления знаний и алгоритмы вывода свойств объектов на основании иерархических, сетевых, фреймовых и других представлений отношений между объектами, описанными функционально, морфологически, атрибутивно и т. д. Для этой цели чаще всего используют объектно-ориентированные языки программирования и специальные языки представления знаний.

5. Проблематика **обработки нечеткой информации и нечеткого вывода** рассматривается в [2]. Следует отметить, что применение нечетких методов ведет к более высокой степени автоматизации сложных, плохо структурированных процессов. Чтобы реализовать нечеткие методы, необходимо иметь соответствующие знания о процессах, которые могут быть адекватно представлены в терминах нечетких правил.

6. **Многоагентные системы** (МАС) – это одно из наиболее молодых направлений интеллектуальных информационных технологий, сформировавшееся в 90-е годы XX века. Задача распараллеливания поиска и обработки информации стала весьма актуальной с появлением таких хранилищ данных, как корпоративные хранилища или сеть Интернет. Основу многоагентных систем составляют автономные агенты – программные агенты, способные воспринимать ситуацию, принимать решения и взаимодействовать с другими агентами. Эти интеллектуальные возможности коренным образом отличают МАС от жестко организованных программных систем, обеспечивают для МАС способность к самоорганизации. При этом отдельные автономные части программы (агенты) получили возможность самостоятельно принимать решения, устанавливать соглашения о том, как будет решаться задача. Агенты приобрели свойство активности и могут вступать в различные отношения между собой, инициировать диалог с пользователем. Можно указать следующие области применения МАС в СПР: интеллектуальный поиск информации (баз данных) в сети Интернет, поддержка проектирования бизнес-процессов, проектирование виртуальных предприятий, составление расписаний.

7. **Интеллектуальный анализ данных** (ИАД), или «добыча данных» (Data Mining) – это процесс аналитического исследования больших массивов информации с целью выявления определенных закономерностей и систематических взаимосвязей между переменными, которые затем можно применить к новым совокупностям данных. ИАД имеет междисциплинарный характер, представляя собой результат взаимодействия идей математической статистики, разведочного анализа данных, машинного обучения, распознавания образов, искусственного интеллекта, технологий баз данных и других дисциплин.

Говоря о больших объемах информации, хранящейся в современных БД, можно привести некоторые примеры. Barclaycard – самая крупная компания Великобритании, работающая с кредитными картами, проводит 350 млн тран-

закций в год. Американская компания розничной торговли Wal-Mart делает свыше 7 млрд транзакций в год. Телефонная компания AT&T выполняет свыше 70 млрд междугородних вызовов в год. Компания MobilOil ставит целью хранить свыше 100 Тбайт данных, связанных с разведкой нефти. NASA спроектировало космическую систему наблюдения за Землей, предназначенную для генерирования порядка 50 Гбайт данных за час в течение столетия. В рамках проекта, связанного с исследованием генома человека, уже собраны гигабайты информации.

Информация, найденная в процессе применения методов «добычи данных», должна быть нетривиальной и ранее неизвестной. Знания должны описывать новые связи между свойствами, предсказывать значения одних признаков на основе других и т. д. Найденные знания должны быть применимы и на новых данных с некоторой степенью достоверности. Алгоритмы, используемые в Data Mining, требуют выполнения большого числа операций, а поэтому они должны обладать достаточным быстродействием. Это обусловило интерес к адаптивным и последовательным методам анализа, которые строят оценки по мере поступления данных, что минимизирует число обращений к дисковой памяти.

Как сказано выше, целью «добычи данных» является поиск структуры. При этом следует различать два типа структур: модели и паттерны (шаблоны). Различие между ними можно охарактеризовать так. *Модель* – это наиболее общий способ описания структуры, справедливый для всего множества данных или для подмножества данных. Нам известны регрессионные модели, модели временного ряда, графовые модели, кластерные модели и т. д. Такая структура представляет полномасштабное описание массы данных.

В противоположность модели, *паттерн* (*шаблон*) – это локальная структура, относящаяся к сравнительно небольшому числу объектов. «Небольшое» может означать и 10, и 100 000. Все зависит от контекста и от объема массива данных. Паттерн определяется относительно модели, поэтому часто является индикатором отклонения от нее, средством выявления аномальных данных (в рамках рассматриваемой модели).

Процесс «добычи данных» отличается от классического *разведочного анализа данных* (РАД): системы **«добычи данных»** в большей степени ориентированы на практическое приложение полученных результатов, чем на выяс-

нение природы явления. При **«добыче данных»** нас не очень интересует конкретный вид зависимостей между переменными задачи. Выяснение конкретной формы многомерных зависимостей между переменными не является главной целью этой процедуры. Основное внимание уделяется поиску решений, на основе которых можно было бы строить достоверные прогнозы.

«Добыча данных» как процесс анализа не может быть одноразовым видом деятельности, которым занимаются спустя длительное время после сбора информации. Это интерактивный процесс, включающий как специалиста в области интеллектуального анализа данных и эксперта предметной области, так и сами данные.

Для решения задач интеллектуального анализа данных широко применяются следующие методы:

1) классификация без обучения (кластер-анализ);

2) классификация с обучением (дискриминантный анализ, деревья классификации);

3) методы прогнозирования (регрессия);

4) временные ряды (выделение тренда и других структур);

5) извлечение правил и закономерностей (поиск ассоциативных правил, деревья решений);

6) нейросетевые методы;

7) графический анализ и визуализация данных.

В последние годы широкое распространение получило новое направление развития ИС Semantic Web. В рамках этого направления создаются новые технологии для размещения ИС в web-пространстве.

3. Средства разработки интеллектуальных систем в сети Интернет

3.1. Средства преставления и хранения знаний в сети Интернет

3.1.1. Онтологический подход. Классификация онтологий

По мнению Т. А. Гавриловой, онтологическое моделирование и онтологический инжиниринг – это методология и технология проектирования, разработки и использования онтологий для структурирования и тиражирования знаний в различных предметных областях и приложениях.

Онтологическое моделирование представляет в настоящее время весьма популярное и быстро развивающееся научно-техническое направление, а полученные здесь результаты активно используются во многих областях информатики.

Научно-технические основы формирования пространств знаний и онтологического моделирования, а затем и онтологического инжиниринга заложили инициатива KA и проект SHOE. Информационные технологии и, в частности web-технологии, тоже внесли и продолжают вносить свой вклад в онтологическое моделирование и онтологический инжиниринг, готовя почву для новых высокотехнологичных продуктов [7].

Формального определения онтологии в настоящее время не существует. В зависимости от конкретных продуктов или научных школ, которые занимаются онтологическим инжинирингом [7], используются разные определения онтологии. В настоящее время наиболее популярным является определение, данное Грубером в 1993 г. По мнению Грубера, онтология – это открытая (явная) спецификация концептуализации – определений понятий, терминов, отношений и механизмов управления, необходимых для описания процессов решения задач в избранной предметной области. Такая спецификация состоит из терминов и правил использования этих терминов, ограничивающих их значения в рамках конкретной области. На формальном уровне онтология – это система, состоящая из набора понятий и набора утверждений об этих понятиях, на основе которых можно строить классы, объекты, отношения, функции и теории. Основными компонентами онтологии являются: классы или понятия, отношения, функции, аксиомы, примеры [8].

В работах [9, 10] рассматриваются различные принципы классификации онтологий, они различаются наборами критериев классификации. В рамках

данного пособия мы остановимся на классификации онтологий по степени формальности – «спектра онтологий», предложенного в работе [8].

Согласно [8] различные классы онтологий могут быть представлены как точки спектра с учетом особенностей их реализации. Каждая точка соответствует наличию ключевых свойств онтологии, отличающих ее от других точек на спектре. Косая черта условно отделяет онтологии от других классов онтологий (рис. 1).

Рис. 1. Спектр онтологий

Контролируемый словарь, соответствующий первой точке на спектре, – это каталог, содержащий однозначную интерпретацию терминов. Такие словари не учитывают омонимию различных терминов предметной области.

Следующая точка на спектре – ***глоссарий*** – список типа «термин – значение». Каждое значение из данной пары описывается на естественном языке. Такое описание дает больше информации, поскольку можно прочесть описание и понять значение термина. Термины могут быть многозначными. В своем обычном виде глоссарии не подходят для автоматической обработки, однако их можно использовать совместно с контролируемыми словарями. При таком подходе каждому термину глоссария ставится в соответствие некоторый идентификатор.

Очередная точка на спектре – ***тезаурусы***. Авторы работы [11] приводят три наиболее популярных определения этого термина:

1) тезаурус – упорядоченный перечень терминов, в котором отражены семантические отношения между ними;

2) тезаурус – свод знаков, терминов, кодов и отношений между ними, которые используются в процессе обмена сведениями, сообщениями;

3) тезаурус – автоматизированный словарь, отображающий семантические отношения между лексическими единицами дескрипторного информационно-поискового языка и предназначенный для поиска слов по их смысловому содержанию.

Среди отношений, свойственных для тезаурусов, можно выделить отношение синонимии, иерархическое отношение и отношение ассоциации.

Первые онтологии, которые нашли применение в сети Интернет, использовались в поисковых системах, основанных на каталогах. В таких онтологиях широко применялось отношение *подкласс – класс* (обозначаемого как ISA). Однако формальные свойства этого отношения не всегда выполнялись. Таким онтологиям на спектре соответствует точка на спектре, подписанная термином ***неформальные таксономии***.

Формальные таксономии следующая точка на спектре. Онтологии этого типа включают точное определение отношения ISA. В таких системах строго соблюдается транзитивность отношения ISA, это необходимо для процедуры логического вывода.

Следующая точка спектра – ***формальные экземпляры***. Данная точка определяет наличие в онтологической системе формального отношения *экземпляр – класс,* обозначаемого как *isInstanceOf*. Некоторые классификации включают только имена классов, другие содержат на нижнем уровне экземпляры (индивиды). Для отношения экземпляр – класс выполняется свойство наследования вдоль отношения ISA: если А является подклассом класса В, то каждый экземпляр класса А также является экземпляром класса В.

Точке ***свойства на основе фреймов*** соответствует появление среди структурных элементов онтологии фреймов и слотов. Здесь классы (или фреймы) могут иметь информацию о свойствах (или слотах).

Следующая точка на спектре – ***ограничения на значения*** – определяет появление в онтологии ограничения на область значений свойств (или слотов).

Значения свойств берутся из некоторого предопределенного множества или из подмножества сущностей онтологии (множество экземпляров данного класса, множество классов). Можно ввести дополнительные ограничения на то, что может заполнять свойство.

Следующие точки на спектре – ***дизъюнктивные классы, обратные свойства*** и ***произвольные логические ограничения*** – связаны с необходимостью более точной спецификации различных свойств сущностей, входящих в онтологию, и поэтому ее структура существенно усложняется. Как правило, первые шаги в этом направлении во многих онтологиях сводятся к введению определения свойств некоторой сущности, входящей в онтологию на основе свойств других сущностей. Многие онтологии позволяют объявлять два и более класса дизъюнктивными (это означает, что у данных классов не существует общих экземпляров). Некоторые языки описания онтологий позволяют делать произвольные логические утверждения о сущностях – аксиомы.

3.1.2. Онтологический подход и Semantic Web

Статья Тима Бернерса Ли [12] в 2001 г. открыла новую веху в истории онтологического подхода – направление Semantic Web, работу над которым начал сам Тим и возглавляемая им организация W3C. В базовую модель Semantic Web авторы последовательно включили [13]:

– спецификации универсальных идентификаторов ресурсов URI (URN + URL), IRI;

– расширяемый язык разметки XML;

– модель описания ресурсов RDF (Resource Description Framework) и языки, обеспечивающие представление RDF-данных;

– средства представления метаданных RDF Schema;

– принципы представления знаний в виде онтологий и языки описания онтологий OWL (OWL Lite, OWL DL, OWL Full), OWL 2;

– язык запросов к RDF-хранилищам SPARQL;

– языки спецификации агентов и сервисов WSDL;

– систему метаданных для регистрации web-сервисов UDDI;

– протокол взаимодействия с сервисами SOAP

и ряд других стандартов.

RDF (Resource Description Framework). Однако еще в 1997 г. консорциум W3C начал работу и определил спецификацию RDF. Концептуально RDF специфицирует правила представления семантических сетей таким образом, чтобы их удобно было хранить и обрабатывать машинам. RDF представляет собой модель для спецификации ресурсов, основанную на описании ресурсов в виде наборов триплетов типа «субъект – предикат – объект». Спецификация RDF опирается на ранние стандарты, лежащие в основе Web: Unicode, URI, XML и XML Schema.

Набор триплетов называется RDF-графом. В качестве вершин графа выступают субъекты и объекты, в качестве дуг – предикаты (или свойства). Направление дуги, соответствующей предикату в триплете, выбирается так, чтобы дуга вела от субъекта к объекту. Предикаты в RDF интерпретируются как бинарные отношения.

На спецификации ресурсов в виде RDF не накладывается никаких ограничений, и они не используют в качестве базовых формализмов какие-либо системы метаданных, что затрудняет машинную обработку этих ресурсов. Для унификации RDF-представлений и описания атрибутов ресурсов и отношений между ними предназначен язык RDF Schema (RDFS) – язык описания словарей для RDF. Все определения RDFS выражены на RDF (поэтому RDF еще называют «самоописывающимся» языком).

Система базовых сущностей RDFS похожа на систему сущностей объектно-ориентированных языков программирования. Основным отличием от объектно-ориентированных языков является то, что в RDFS центральным аспектом является определение отношения, а не класса или объекта. Отношения в RDF определяются как пары (домен – диапазон). При этом домен определяет множество RDF-классов, к которым данное отношение может быть применено, диапазон определяет множество ресурсов – значений отношения (свойства). В большинстве объектно-ориентированных языков, за исключением языков, подобных Python или Ruby, определение класса имеет фиксированную форму – свойства класса выражаются в полях и методах класса. В RDF, напротив, набор отношений класса определяется вне самого класса.

Пример. Определим отношение «изучать» с доменом «Студент» и диапазоном «Дисциплина». Если необходимо расширить определение классов «Студент»

и «Дисциплина», то можно определить новое отношение с соответствующим ему доменом:

Класс *Дисциплина*; ***Класс*** *Студент*;

Свойство *изучать (Дисциплина*, *Человек)*.

Вывод: В большинстве объектно-ориентированных языков определение отношения (свойства) класса является неотъемлемой частью его определения. Таким образом, изменение семантики отношения приводит к изменению семантики класса. В RDFS же при изменении смысла отношения изменять придется именно их. При этом все классы, зависящие от изменяемых свойств, косвенно изменят свою семантику.

Зачастую при спецификации предметной области необходимо формулировать утверждения о других утверждениях, в RDFS для этих целей используется механизм реификации.

Наиболее важными свойствами RDF являются:

• обобщенный способ работы с метаданными;

• ориентация на программное обеспечение в качестве конечного потребителя информации.

Недостатки RDF. Основные достоинства RDF – открытость и расширяемость одновременно являются и его основными недостатками. Открытость и расширяемость позволяют любому пользователю RDF фиксировать произвольное утверждение о чем угодно. Не запрещается делать бессмысленных утверждений или утверждений, не согласующихся с другими. Вся ответственность за проверку корректности сделанных утверждений возлагается на разработчиков стандартов метаданных или приложений обрабатывающих RDF-данные.

Язык OWL (Web Ontology Language) является одним из способов представления знаний, получивших в последние годы наибольшее распространение и является еще одним результатом работы в рамках инициативы Semantic Web. Инициаторами разработки и стандартизации выступили авторы разработанных ранее онтологических языков DAML (DARPA Agent Markup Language) и OIL (Ontology Inference Layer).

OWL фактически является надстройкой над RDF/RDFS и поддерживает эффективное представление онтологий в терминах классов и свойств, обеспечение простых логических проверок целостности онтологии, связывание онтологий друг с другом (импорт внешних определений). Большое число создавае-

мых в настоящее время онтологий кодируются на OWL, уже существующие онтологии транслируются в него. По сравнению с RDFS OWL предоставляет дополнительные возможности: определение класса через ограничения на его свойства, через теоретико-множественные операции над классами, определение характеристик (симметричность, транзитивность и др.) и кардинальных чисел свойств.

В настоящее время существуют две версии языка OWL: OWL 1, включающий три диалекта разной выразительной мощности (OWL Lite, OWLDL, OWL Full), и OWL 2.

Недостатки OWL. Одним из существенных недостатков OWL является отсутствие возможности естественным образом определять свойства у свойств. Это не позволяет моделировать атрибуты у предметных отношений, *n*-арные отношения и атрибуты у атрибутов[14].

Пусть, например, в онтологии зафиксирован факт встречи двух людей:

<человек rdf:ID="Петров">

<встретился_crdf:resource="#Сидоров"/>

</человек>

Если мы теперь захотим зафиксировать дату встречи, то синтаксис OWL не позволит нам это сделать.

Существует два пути обхода указанной проблемы. Использовать возможности RDFS и прибегнуть к механизму реификации (от англ. reify – материализовывать). Ее суть состоит в том, чтобы рассматривать RDF-утверждения как отдельные объекты. Альтернативный путь – декларировать отношения как классы, а не как свойства. Авторы этого решения критикуют использование реификации для определения свойств у свойств, приводя следующие доводы. «В *n*-местных отношениях дополнительные аргументы обычно характеризуют не само утверждение (statement), а экземпляр отношения». Поэтому было бы более естественным оперировать непосредственно экземплярами отношений, а не утверждениями об этих экземплярах [14].

3.2. Другие средства разработки онтологий и онтологические модели

Проект CYC – Cyc (Имя «Cyc» представляет собой выдержку из английского «enCYClopedia» и произносится как «Цик») представляет собой программную систему, библиотека онтологий которой создана для описания зна-

ний, необходимых для рассуждений на бытовом уровне, т. е. таких знаний, которые человек использует в своей каждодневной деятельности. Разработка Cyc началась в 1984 г. под руководством Дугласа Лената в компании Microelectronicsand Computer Technology Corporation (MCC). В 1994 г. была учреждена компания Cycorp, целью которой являлась коммерческая деятельность в области, связанной с проблематикой искусственного интеллекта.

Знания в библиотеке онтологий Cyc поделены на микротеории. Микротеория содержит факты, касающиеся той или иной области знаний. И представляет собой аналог онтологии. Для записи фактов в микротеориях Cyc использует специальный язык под названием CycL (Cyc Language). Язык CycL представляет собой язык исчисления логики предикатов первого порядка с аксиомами эквивалентности, возможностями расширения процедур логического вывода и сколемизацией (сколемизация – это процесс устранения кванторов существования путем их удаления). Язык также поддерживает некоторые свойства языка второго порядка (например, квантификацию по предикатам с некоторыми ограничениями).

База знаний Cyc состоит из термов (terms) – словаря языка CycL и суждений (assertions), связывающих эти термы. Система Cyc используется не только, как хранилище для суждений, но также предоставляет возможности по осуществлению логического вывода по этим суждениями. Библиотека онтологий проекта Cyc довольно обширна. Так известно, что еще в 1994 г., когда была создана компания Cycorp, библиотека содержала порядка 250 тыс. фактов. На данный момент, онтология Cyc содержит более 2 млн суждений. Для систематизации такого обширного комплекса знаний в Cyc используется своя онтология верхнего уровня [15].

Система логического вывода Cyc (Cyc Inference Engine) поддерживает выводы на основе модус поненс и модус толленс, использование кванторов всеобщности и существования, а также математические преобразования. Логический вывод производится в контексте микро теорий, позволяя таким образом ограничивать рассуждение суждениями, относящимися только к данной микротеории и ко всем ее родительским микротеориям. Система Cyc позволяет расширять встроенные процедуры логического вывода новыми модулями. Таких модулей достаточно много [16].

Проект WonderWeb. ***Онтология верхнего уровня DOLCE*** первая из онтологий в библиотеке проекта WonderWeb (http://www.loa-cnr.it/DOLCE.html), которая была разработана для Semantic Web с целью согласования «мнений» между интеллектуальными агентами, использующими разную терминологию. Основная цель разработчиков DOLCE состояла в том, чтобы создать модель, помогающую при сравнении и объяснении связей с другими онтологиями библиотеки WFOL (базовой библиотеки онтологий WonderWeb), а также для выявления скрытых допущений, лежащих в основе существующих онтологий и лингвистических ресурсов, таких, как WordNet.

В основу проектирования онтологии DOLCE положено фундаментальное философское разделение всех сущностей на универсалии (сущности потенциально или реально имеющие экземпляры) и индивиды (или частности), которые не имеют и не могут иметь экземпляров, и при этом DOLCE – онтология индивидов, в том смысле что область описания ограничена только ими. Еще одна особенность онтологии DOLCE – явное разделение на «Постоянные» и «Происходящие» сущности, различие между которыми состоит в том, что первые имеются в наличии целиком и неизменно в некотором фиксированном промежутке времени (например, компания в течение времени своего существования), а вторые разворачиваются во времени, и в каждый момент в некотором временном интервале они могут быть различными (например, ураган), но при этом сохранять свою идентичность. Такое разделение на «объект» и «процесс» весьма условно и привело, например, к тому, что в онтологии определены два типа отношения «ЧАСТЬ – ЦЕЛОЕ» (первое никак не зависит от времени, второе имеет временной индекс, определяющий в каких временных рамках отношение действует).

Для представления своей онтологии авторы DOLCE избрали более гибкий, чем в проекте Cyc, подход. Здесь онтология сначала фиксируется на бумаге с использованием исчисления предикатов первого порядка, затем для нее описывается та часть аксиом, которая может быть представлена на языке OWL, а оставшиеся аксиомы, выраженные на языке KIF [17], добавляются к OWL-описаниям в виде комментариев.

Таким образом, достигается выразительность уровня KIF и совместимость с OWL, хотя приложения, не имеющие информации о действительной

структуре OWL-документа, не смогут получить доступ к «закомментированным» знаниям.

Следует отметить, что в библиотеке онтологий WonderWeb в настоящее время имеется спектр онтологий разного уровня для разных предметных областей. И в этом смысле онтология DOLCE жива и используется на практике [8, 18].

SUMO (Suggested Upper Merged Ontology) – онтология верхнего уровня, разработанная в рамках проекта рабочей группы IEEE SUO (IEEE Standard Upper Ontology Working Group) и Teknowledge SUOWG в 2003 г.

Онтология SUMO содержит наиболее общие и самые абстрактные концепты, имеет исчерпывающую иерархию фундаментальных понятий (около тысячи) и впечатляющий набор аксиом (примерно четыре тысячи), определяющих эти понятия. SUMO направлена на содействие улучшению интероперабельности данных, извлечению и поиску информации, автоматическому выводу (доказательствам), обработке естественного языка и др. Онтология охватывает следующие предметные области: общие виды процессов и объектов, абстракции (теория множеств, атрибуты, отношения), числа и единицы измерения, временные понятия, части и целое, агенты и намерения.

SUMO содержит обозримое число концептов и аксиом, имеет ясную иерархию классов, легко расширяется, является итогом объединения различных общедоступных онтологий верхнего уровня, что характеризует ее тоже как онтологию верхнего уровня.

К преимуществам SUMO можно отнести возможность трансляции описания онтологии на любой из основных языков представления знаний, наличие онтологии среднего уровня (MILO), «бесшовно» интегрированной с SUMO, несколько десятков примеров практического применения, а также связь с WordNet [19] – наиболее известным на данный момент тезаурусом, в котором содержится около 150 тыс. слов разговорного английского языка.

Основными концептами SUMO являются «Сущность» и ее категории – «Физический» и «Абстрактный». Первая категория включает все, что имеет положение в пространстве – времени, а вторая – все остальное (а точнее, только то, что существует в воображении). Категория «Физический» делится на «Объект» и «Процесс», что соответствует подходу, реализованному в DOLCE.

Онтология SUMO активно используется в разных прикладных проектах, где применяются онтологические модели [8, 18].

Онтология Джона Совы, предложенная в работе [20], определяет базовые онтологические категории логики, лингвистики, философии и искусственного интеллекта. По мнению Совы онтология верхнего уровня для сохранения «открытости» должна быть основана не на фиксированной иерархии концептов, а на каркасе, описывающем различия, по которому иерархия генерируется автоматически. Джон Сова разработал онтологические категории верхнего уровня. Кроме категорий в онтологии Дж. Совы введены следующие понятия: «Сущность», которое не определяет никаких отличительных признаков или различий и является надтипом для всех других концептов; «Абсурдный» – тип, наследующий все возможные, в том числе противоречащие различия, у которого нет ни одного экземпляра. Кроме того, в онтологии Дж. Совы проводится различие между абстрактным и физическим (именно в таком виде оно заимствовано разработчиками SUMO), а отдельно выделяются категории независимости, относительности и опосредованности.

«Независимые» сущности не нуждаются в существовании каких-либо связей с другими сущностями, но любая «Относительная» сущность обязательно имеет хотя бы одну связь с некоторой другой сущностью. Для существования «Опосредованной» сущности необходимо наличие некоторого отношения, связывающего какие-то другие сущности, имеющие отношение также и к первой (например партнерство).

Онтология Джона Совы описывает роли и отношения, агентов, процессы, и т. д. Заметим, что онтология Дж. Совы достаточно часто используется в проектах в качестве верхней онтологии [8, 18].

3.2.1. Семантическая разметка и семантическое аннотирование в web-пространстве

Современные web-системы предоставляют пользователям средства простого добавления и редактирования содержимого: все, что требуется от пользователя – это знание разметки и наличие браузера. Рассмотренные средства представления знаний в web-пространстве не всегда ориентированы на пользователя, знакомого с традиционными средствами разметки, и требуют наличия специальной подготовки. Кроме этого, для многих прикладных задач не всегда

требуется использование возможностей глубоко формального представления предметной области. В виду названных причин широкое применение в промышленной разработке программного обеспечения для web-пространства получили средства семантического аннотирования и средства семантической разметки web-документов.

Для удобства изложения материала ***семантическим аннотированием*** будем называть придание фрагментам некоторого неформального текста семантической (формальной) значимости в рамках web-системы. А ***семантической разметкой*** назовем расширения языка HTML, XHTML или XML, позволяющие определять на страницах, представленных с помощью этих языков, семантически значимые сущности.

В настоящее время существуют следующие наиболее употребимые средства семантической разметки: микроформаты, RDFa и микроданные.

Микроформаты (microformats, μF) – это способ семантически размечать сведения о разнообразных сущностях на web-страницах, используя стандартные элементы языка HTML (или XHTML). В настоящее время микроформаты используются как для унифицированного представления однотипных сущностей в web-приложениях, так и для настройки и адаптации пользовательского интерфейса web-браузеров к пользователю.

Наиболее известные микроформаты:

– hCard – для публикации контактной информации людей, компаний, организаций и мест;

– hCalendar – микроформат для представления семантической информации о событиях в формате календаря;

– hAtom – ленты новостей;

– XFN – социальные взаимоотношения.

Приведем пример использования микроформата hCard:

– HTML, описывающий контактную информацию человека:

```
<div>
     <div>Иванов Иван Иванович</div>
     <div>21.12.1988</div>
     <div>ООО "Example"</div>
     <div>604-55-14</div>
     <ahref="http://example.com/">http://example.com</a>
     <ahref="mailto:info@example.com">info@example.com</a>
</div>
```

– с добавлением микроформатов выглядит так:

```
<divclass="vcard">
     <divclass="fn">Иванов Иван Иванович</div>
     <div class="bday">21.12.1988</div>
     <div class="org">ООО "Example"</div>
     <div class="tel">604-55-14</div>
     <a
class="url"href="http://example.com/">http://example.com</a>
     <a
class="email"href="mailto:info@example.com">info@example.com<
/a>
</div>
```

Микроданные или проект Schema.org – это стандарт семантической разметки данных в сети, объявленный поисковыми системами Google, Bing и Yahoo летом 2011 г. С осени 2011 г. к этому проекту присоеденилась российская компания Yandex.

Цель семантической разметки – сделать Интернет более понятным, структурированным и облегчить поисковым системам и специальным программам извлечение и обработку информации для удобного ее представления в результатах поиска. Разметка происходит непосредственно в HTML-коде страниц с помощью специальных атрибутов и не требует создания отдельных экспортных файлов. По сравнению с микроформатами в микроданные добавлены возможности расширения существующих типов данных, появилась возможность указывать типы сущностей и экземпляры сущностей, в отличии от микроформатов, где можно было указывать лишь экземпляры сущностей.

В настоящее время разработано более 100 связанных между собой типов микроданных, приведем некоторые:

– творческие произведения: Creative Work (творческое произведение), Book (книга), Movie (фильм), Music Recording (музыкальная запись), Recipe (рецепт), TV Series (телесериал) и др.;

– встроенные нетекстовые объекты: Audio Object (аудио), Image Object (изображение), Video Object (видео) и др.;

– Event (событие);

– Organization (организация);

– Person (человек);

– Place (место), Administrative Area (административная единица), Local Business (местная фирма), Restaurant (ресторан) и др.;

– Product (продукт), Offer (предложение), Aggregate Offer (сводное предложение) и др.;

– Review (отзыв), Aggregate Rating (сводный рейтинг).

Приведем пример оформления с помощью микроданных:

– HTML, описывающий контактную информацию человека:

```
<div>
     <div>Иванов Иван Иванович</div>
     <div>21.12.1988</div>
     <div>ООО "Example"</div>
     <div>604-55-14</div>
     <ahref="http://example.com/">http://example.com</a>
     <ah-
ref="mailto:info@example.com">info@example.com</a>
</div>
```

– с добавлением микроданных выглядит так:

```
<div itemscopeitemtype="http://data-
vocabulary.org/Person">
     <divitemprop="name">Иванов Иван Иванович</div>
     <divitemprop="birthDate">21.12.1988</div>
     <div itemprop="affiliation">ООО "Example"</div>
     <div itemprop="telephone">604-55-14</div>
     <a item-
     prop="url"href="http://example.com/">http://example
     .com</a>
     <a item-
     prop="email"href="mailto:info@example.com">info@exa
     mple.com</a>
</div>
```

RDFa-разметка – технология от W3C, которая представляет собой способ разметки содержания для описания специального типа данных в виде RDF-графа, представленного с помощью атрибутов HTML или XHTML. Эти типы данных указываются сущностями или элементами. Каждая сущность имеет ряд свойств.

В общем случае RDFa использует простые атрибуты в тегах XHTML (обычно или) для назначения кратких описательных названий сущностей и их свойств. Ниже приведен пример небольшого HTML-блока, показывающего использование RDFa-разметки:

– HTML, описывающий контактную информацию человека:

```
<div>
```

```
      <div>Иванов Иван Иванович</div>
      <div>21.12.1988</div>
      <div>ООО "Example"</div>
      <div>604-55-14</div>
      <ahref="http://example.com/">http://example.com</a>
      <ahref="mailto:info@example.com">info@example.com</a>
</div>
```

– с добавлением RDFa выглядит так:

```
<div divxmlns:v="http://rdf.data-
vocabulary.org/#"typeof="v:Person">
      <divptoperty="v:name">Иванов Иван Иванович</div>
      <divptoperty="v:birthDate">21.12.1988</div>
      <div ptoperty="v:affiliation">ООО "Example"</div>
      <div ptoperty="v:telephone">604-55-14</div>
      <a ptoper-
      ty="v:url"href="http://example.com/">http://example.com<
      /a>
      <a ptoper-
      ty="v:email"href="mailto:info@example.com">info@example.
      com</a>
</div>
```

Наиболее часто семантическое аннотирование используется в web-проектах, построенных на базе wiki-технологий. Исторически сложилось, что в таких проектах развивается два типа семантического аннотирования: на основе инструмента wiki-шаблонов (примеры использования можно увидеть в рамках проекта wikipedia.org) и на основе расширения базового набора тегов wiki-системы (примеры использования – в проектах semanticweb.org, Semantic Mediawiki, Enterprise Semantic Mediawiki (SMW+)).

Подход, который использован в проекте ru.wikipedia.org, очень близок к подходам, использованным в различных версиях семантических разметок, рассмотренных ранее. Разработчиками определены ключевые сущности и их свойства и для них разработаны wiki-шаблоны, с помощью которых участники проекта Wikipedia должны оформлять статьи. Пример оформления семантических аннотаций (фрагмент статьи «Компьютерные науки»), использующихся на сайте ru.wikipedia.org:

*Также существует дискуссия, считать ли [[Разработка программного обеспечения|разработку программного обеспечения]] частью компьютерных наук или нет****<ref>{{cite journal***

| last = Parnas

| first = David L.

| author link = David Parnas

| year = 1998

| title = [http://citeseer.ist.psu.edu/parnas98software.html Software Engineering Programmes are not Computer Science Programmes]

| journal = Annals of Software Engineering

| volume = 6

| pages = 19–37

| doi = 10.1023/A:1018949113292

}}, p. 19: «Rather than treat software engineering as a subfield of computer science, I treat it as an element of the set, Civil Engineering, Mechanical Engineering, Chemical Engineering, Electrical Engineering, ..»</ref>.

В отличие от обычных wiki-систем Semantic Mediawiki позволяет семантически аннотировать содержимое сайта в том числе и на основе разработанных внешних онтологий. Эта модель аннотирования также гибка и легко расширяема. Кроме того, она не требует никакого предварительного знания определенной заранее системы метаданных – эта система может быть использована в системе позже. Примеры использования семантических аннотаций в Semantic Mediawiki приведены в табл. 1.

Таблица 1

Семантические аннотации Semantic Mediawiki

Что делает	**Как выглядит**
Связывает сущность, описываемую на странице, с числом 1 234 567 свойством «население»	В этой стране население [[население::1 234 567]] человек
Связывает сущность, описываемую на странице, с числом 999 331 свойством «население» и вместо числа запись «около миллиона»	В этой стране [[население::999 331|около миллиона]]

В wiki-системах в качестве основного инструмента используются шаблоны и формы ввода, что позволяет задать ограничения на используемое множество видов семантических аннотаций.

Шаблоны задают логику и внешний вид части страницы. Внутри шаблона используются переменные, значения которых подставляются при вызове шаблона на странице. Если в тексте шаблона встречаются семантические свойства, то аннотируются все страницы, использующие этот шаблон. Следовательно,

при изменении значений свойств в шаблоне каскадно обновляются все зависимые от него страницы.

Формы ввода предоставляют графический интерфейс для заполнения шаблонов корректными значениями, а их использование не требует знания wiki-разметки. Таким образом, сочетание возможностей шаблонов и форм позволяет иметь множество определенных заранее видов аннотаций и обеспечить унификацию представления.

3.2.2. Хранилища знаний

Работы, связанные с организацией хранилищ знаний для промышленного программного обеспечения, в настоящее время ведутся в основном для знаний, представленных в виде RDF. Поэтому мы рассмотрим основные подходы, которые используются сегодня для организации хранилищ RDF-графов. Перечислим эти подходы:

– загрузка всего графа в оперативную память (использовано в проектах: cwm, Python RDFLib);

– специализированные СУБД (использовано в проектах: Redland, Sesame, DOGMA);

– таблица триплетов в реляционной СУБД – все утверждения хранятся в одной таблице, включающей поля «субъект», «предикат» и «объект» (использовано в проектах: Jena TDB, KAON, RDF-3X);

– отображение реляционной базы данных приложения на модель данных RDF (использовано в проектах: Federate, D2QR, Virtuoso, Graffity).

Рассмотрим вкратце каждый из подходов. Загрузка графа в оперативную память использовалась на ранних стадиях развития RDF для концептуального прототипирования и, как правило, не используется для долгосрочного хранения больших объемов данных.

СУБД, разработанные специально для хранения RDF-данных, используют сетевую (графовую) или иерархическую модели данных. В долгосрочной перспективе данных подход может оказаться наиболее эффективным благодаря более полному соответствию графовой модели данных RDF по сравнению с реляционными СУБД. Однако он обладает целым рядом недостатков с точки зрения практики:

1. Использование специализированной БД требует переноса данных из существующих баз данных, которые в подавляющем большинстве используют реляционные СУБД.

2. Технологическое отставание сетевых СУБД от реляционных, обусловленное повсеместным использованием реляционных БД в последние 30 лет.

Преимущества реляционных СУБД обусловили широкое распространение подхода, основанного на хранении RDF-графов в реляционных базах данных в виде таблицы триплетов. Данный подход ориентирован в первую очередь на решение проблемы технологического отставания специализированных СУБД для хранения RDF от реляционных. Однако и этот подход не лишен недостатков, которые связаны в первую очередь:

1) с проблемой переноса данных;

2) с тем, что на больших наборах данных таблица триплетов существенно снижает скорость обработки запросов;

3) с тем, что в случае, когда реляционная схема достаточно точно отражает предметную область, требования производительности и масштабируемости могут перевешивать гибкость, обеспечиваемую более универсальной таблицей триплетов.

В случае отображения реляционной базы данных приложения на модель данных RDF для существующей реляционной БД строится надстройка, которая позволяет работать с такой базой как с RDF-данными, используя специальный язык запросов (SPARQL, RQL и др.). Сегодня данный подход показывает более высокую производительность для промышленных систем, чем рассмотренные выше.

Рассмотрим наиболее популярные в настоящее время хранилища.

RDF-хранилище Sesame использует подход, в основе которого лежит идея независимости от платформы хранения, позволяющая не заботится о том, какая модель данных используется для представления информации на нижнем уровне. Слой, который в рамках проекта Sesame обеспечивает независимость от платформы, назван «the Storage And Interface Layer» или сокращенно – SAIL [21]. Этот слой является интерфейсом прикладного программирования (API), предоставляющим клиентским приложениям специфичные для RDF методы, которые, в свою очередь, транслируются в запросы к соответствующим СУБД [21]. В рамках

Sesame поддерживается возможность использования SPARQL [22], RQL, SeRQL и RDQL языков запросов. Манипуляции над данными в Sesame производятся с помощью «Sesame's Access APIs», который, в свою очередь, делится на «the Repository API» и «the Graph API». Первый предоставляет высокоуровневый доступ к репозиториям, обслуживаемым Sesame, обеспечивающим выполнение запросов, хранение RDF-файлов, извлечение RDF и т. д. Второй предоставляет достаточно детальную поддержку манипуляций над RDF-данными, например, добавление и удаление отдельных утверждений, создание малых RDF-моделей напрямую из программного кода. Интерфейсы приложения Sesame дополняют друг друга в функциональном плане и на практике используются совместно. Доступ к «Sesame's Access APIs» можно получить как из приложений, находящихся на локальном компьютере с использованием Sesame в качестве библиотеки, так и с удаленной станции, воспользовавшись каким-либо из HTTP протоколов: Remote Method Invocation (RMI) или the Simple Object Access Protocol (SOAP).

Хранилище HyperGraphDB расширяет понятие гиперграфа, позволяя ребру указывать на другое ребро, и дает возможность любому ребру или вершине иметь произвольное значение в виде полезной нагрузки [23]. Основная единица хранения в HyperGraphDB называется атомом. Каждый атом имеет произвольное значение и может указывать любое количество других атомов. Типы данных управляются единой расширяемой системой, встроенной в структуру гиперграфов. Схема хранения данных платформенно независимая и позволяет работать с ними с помощью любого языка программирования на любой платформе. Хранилище HyperGraphDB не вносит ограничений на размер графа, но размер каждой отдельной сущности ограничен системой. Система HyperGraphDB не поддерживает языков запросов, однако предоставляет высокоуровневое API, которое позволяет описывать поисковые шаблоны для видов графов, поддерживаемых в системе.

Хранилище Neo4j является сетевой СУБД. Для представления данных использует модель, расширяющую традиционную модель семантической сети: задаются сущности, отношения между сущностями и их свойства; вводятся специальные типы элементов узлов и дуг, которые могут хранить индексы сущностей и отношений. Одновременно с такими элементами возможны операции и с обычными сущностями, и с отношениями. В Neo4j механизмы навигации доступны посредством простого API и с помощью языка запросов Cypher, который позволяет задавать простейшие поисковые шаблоны для графов [24].

В состав ***хранилища AllegroGraph*** включен фреймворк, использующий хранилища на основе дискового пространства. Хранилище AllegroGraph может подключать хранилища Sesame, Jena, Clojure в качестве платформ нижнего уровня. Для обеспечения эффективного семантического поиска в AllegroGraph реализовано два основных типа вывода на знаниях: RDFS++ (с поддержкой следующих предикатов: RDF:type, RDFs:subClassOf, RDFs:range, RDFs:domain, RDFs:subpropertyof, owl:sameAs, owl:inverseOf, owl:TransitiveProperty, owl:hasValue) и RacerPro – верификатор, который имеет выразительность OWL DL [25].

В состав ***хранилища Virtuoso Universal Server*** входит одно из самых производительных и масштабируемых RDF-хранилищ. Данное хранилище снабжено гибкой технологией SPARQL-endpoint с поддержкой RDF Views. RDF Views – это реализация RDBMS2RDF, позволяющая динамически работать с реляционными данными через SPARQL-запросы. Встроенный верификатор позволяет поддерживать вывод на знаниях, используя предикаты RDFs:subClassOf, RDFs:subPropertyOf, owl:equivalentClass, owl:equivalentProperty и owl:sameAs. Кроме различных функциональных API, поддерживаются также и различные языки запросов: SQL, SPARQL, XQuery [26].

3.3. Средства поиска знаний

3.3.1. Организация поиска в хранилищах знаний

В рамках данного раздела будут рассмотрены принципы организации поиска знаний, представленных в виде RDF-графов. В основу современных машин поиска, использующихся в RDF-хранилищах, положено решение задачи нахождения подграфа, изоморфного данному графу. И хотя указанная задача, сформулированная в общем виде, относится к классу NP-полных, применительно к RDF-графам она существенно упрощается за счет того, что на практике часть элементов искомого RDF-подграфа всегда известна.

Схема работы типовой поисковой машины поиска для RDF-хранилища представлена на рис. 2.

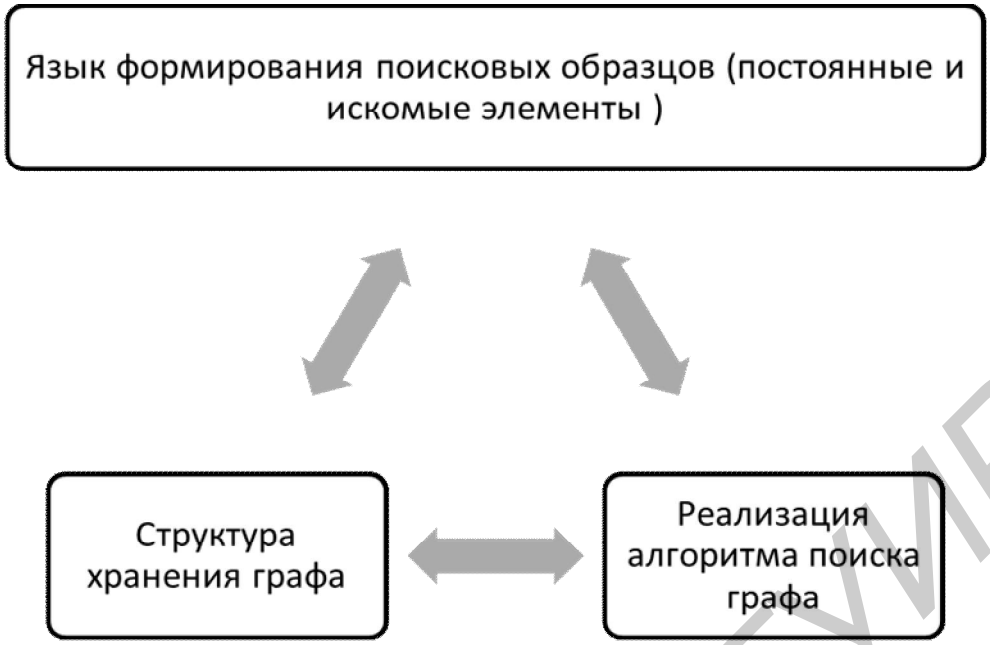

Рис. 2. Упрощенная схема работы поисковой машины

Поисковая машина должна иметь языковые средства описания поисковых образов (формулировки поисковых запросов). Как правило, такие языковые средства ориентированы на работу с информационной структурой или структурами определенного вида, зависящими от специфики реализации хранилища. Реализация алгоритма поиска в зависимости от подходов, использованных для представления RDF-графа на нижнем уровне, может быть с элементами логического вывода по определенному набору предикатов или без реализации логического вывода.

Как правило, в языковые средства описания поисковых образов включают средства описания шаблонов триплетов (обычно это триплет с переменной обозначаемой символом «?»). ***Базовым графовым шаблоном (БГШ)*** называют набор шаблонов триплетов, такой, что $\boldsymbol{bgp = (tp_1, tp_2, \ldots, tp_n)}$, где каждый tp_i является шаблоном триплетов. На шаблонах триплетов определяют следующие операции объединения шаблонов:

– операцию на базе семантики связывания (*join* semantics), если не задана операция между БГШ;

– операцию на базе семантики левого соединения (*left join* semantics), если задано ключевое слово OPTIONAL;

– операцию на базе семантики объединения (*union* semantics), если задано ключевое слово UNION.

Для формирования поисковых запросов в языковых средствах формирования поисковых образов определяют следующие типы поисковых запросов:

– запрос для выбора конкретных элементов (значения вершин или ребер RDF-графа), в результате выполнения получается набор (bag) решающих соответствий (solution mappings);

– запрос получения любой информации о ресурсах, которые были отобраны с помощью шаблона графа в форме другого RDF-графа;

– запрос для построения другого (alternative) графа на основе использования шаблона конструкции (construct template), который заполняется решениями из шаблона графа;

– запрос для проверки логического вывода параметризованных подграфов, возвращающий true, если существуют возможные решения, и false – в противном случае.

Наиболее популярными среди языков запросов к RDF-хранилищам в настоящее время являются языки RDQL и SPARQL. Рассмотрим несколько упрощенный синтаксис SPARQL-запроса и пример его работы для демонстрации схемы работы поискового механизма в RDF-хранилище:

SELECT <v_list>
FROM <onologyURI>
WHERE { <template_list>.
FILTER <filter_expr>
}

- *v_list – список имен переменных*
- *onologyURI – URI-ссылка на онтологию*
- *template_list – список шаблонов*
- *filter_expr – ограничения на значения переменных*

Допустим, онтология содержит следующие RDF-триплеты:

(Foo1, category, “Total Members”);
(Foo1, rdf:value, 199);
(Foo2, category, “Total Members”);
(Foo2, rdf:value, 200);
(Foo2, category, “CATEGORY X”);
(bar, category, “CATEGORY X”);
(bar, rdf:value, 358).

Проследим за ходом выполнения запроса (имена переменных предваряются знаком«?»):

SELECT ?cat ?val WHERE {?x rdf:value ?val. ?x category ?cat. FILTER (?val>=200).}

Семантика запроса: выдайте все объекты cat предиката category, субъект которого (*x*) является также субъектом предиката rdf:value со значением val, не меньшим 200. Вместе со значениями cat выдайте соответствующие значения val.

Ход выполнения запроса: на место переменной *x* могут быть подставлены Foo1, Foo2 и bar (из исходной онтологии), причем Foo2 может быть подставлен дважды, поскольку имеет два свойства category. При подстановке Foo1 значение переменной val не удовлетворяет ограничению в предложении FILTER. Во всех остальных случаях все условия запроса выполнены. Ниже представлен результат выполнения запроса.

Результат выполнения запроса: 3 пары значений (cat, val)

[
["Total Members", 200],
["CATEGORY X", 200],
["CATEGORY X", 358]
]

Недостатки SPARQL. Язык запросов является одним из важнейших аспектов работы с инструментарием, т. к. именно неудачный компромисс между удобством и функциональной мощью языка может скомпрометировать продукт в глазах основной потребительской массы.

Анализ используемых в настоящее время языков запросов для работы с RDF-хранилищами показывает [27–29], что:

– SPARQL позволяет задавать только примитивные шаблоны поиска по графовому образцу, эквивалентные искомому подграфу с точностью до изоморфизма, однако для формулировки семантических запросов этого бывает явно недостаточно;

– SPARQL не позволяет использовать выразительные средства OWL и RDFS при описании графового шаблона – это приводит к тому, что в рамках конкретных RDF-хранилищ приходится реализовывать платформенно-зависимые процедуры логического вывода, как это сделано в хранилищах AllegroGraph, Virtuoso и Sesame, чтобы частично или полностью исключить данный недостаток;

– для задания графового шаблона в SPARQL используются вложенные подзапросы, что может привести к серьезному снижению эффективности поиска исходя из опыта работы с SQL.

Для устранения приведенных недостатков, а также для повышения эффективности выполнения запросов такие RDF-хранилища вынуждены использовать свои языки запросов, аналогичные SPARQL по характеристикам, но оптимизированные под конкретную реализацию хранилища RDQL (Jena, Sesame, PHPxmlclasses, RDFStore), iTQL [30] (Mulgara), RQL (RDF querylanguage для PostgreSQL) [31], RQL [32] (RelationQuery Language (для проекта CubicWeb)), SeRQL (язык Prolog и Sesame), Versa [33] (для проекта 4Suite XML framework).

4. Открытая семантическая технология компонентного проектирования интеллектуальных систем (технология OSTIS)

4.1. Принципы технологии OSTIS

В настоящее время актуальной проблемой в области искусственного интеллекта является разработка технологий, позволяющих быстро создавать разнообразные прикладные интеллектуальные системы. Основными составляющими таких технологий являются:

– формальная теория интеллектуальных систем (формальное описание того, как они устроены);

– методы проектирования интеллектуальных систем;

– инструментальные средства (средства автоматизации проектирования интеллектуальных систем);

– средства информационной поддержки (информационного обслуживания) разработчиков интеллектуальных систем;

– средства компьютерной поддержки управления коллективной разработкой интеллектуальных систем.

Современные технологии проектирования интеллектуальных систем имеют ряд недостатков [34]:

1. Технологии искусственного интеллекта не ориентированы на широкий круг разработчиков интеллектуальных систем и не получили массового распространения.

2. Велики сроки разработки интеллектуальных систем и трудоемкость их сопровождения.

3. Высока степень зависимости технологий искусственного интеллекта от платформ, на которых они реализованы.

4. Отсутствуют подходы, позволяющие на некоторой универсальной основе интегрировать научные и практические результаты в области искусственного интеллекта, что порождает высокую степень дублирования результатов.

5. Эффективная реализация существующих моделей представления знаний и моделей решения трудно формализуемых задач сдерживается техническими возможностями современных компьютеров.

Голенковым В. В. предложен подход, направленный на создание массовой технологии быстрого проектирования интеллектуальных систем. Этот подход характеризуется следующим:

- ориентация на семантическое представление знаний, которое полностью абстрагируется от особенностей технической реализации интеллектуальных систем;
- унификация моделей интеллектуальных систем, направленная на обеспечение их интегрируемости;
- модульное (компонентное) проектирование на основе библиотек типовых многократно используемых компонентов интеллектуальных систем;
- полная совместимость инструментальных средств проектирования с проектируемыми системами – инструментальные средства строятся как интеллектуальные системы на основе тех же принципов;
- включение в состав технологии проектирования интеллектуальных систем комплексной интеллектуальной help-системы для разработчиков интеллектуальных систем;
- включение в состав проектируемых интеллектуальных систем help-подсистем, ориентированных на повышение квалификации конечных пользователей;
- включение в состав проектируемых интеллектуальных систем подсистем самотестирования (самодиагностики, самоанализа) и подсистем, ориентированных на автоматическое или максимально автоматизированное повышение собственного качества.

Технологии проектирования интеллектуальных систем не зависят от многообразия вариантов технической реализации как самой технологии, так и проектируемых интеллектуальных систем и компьютерных архитектур, специально ориентированных на обработку баз знаний. Решение рассматриваемой проблемы предполагает:

– логико-семантический уровень рассмотрения технологии и проектируемых интеллектуальных систем, абстрагирующийся от деталей их технической реализации;

– автоматический переход от разработанного абстрактного логико-семантического уровня проектируемой интеллектуальной системы к различным вариантам ее технической реализации.

Указанный логико-семантический уровень (логико-семантическая модель) интеллектуальной системы должен быть основным объектом внимания проектировщика интеллектуальной системы. Результатом работы на этом уровне является полное описание проектируемой интеллектуальной системы, в котором рассматриваются вопросы, связанные с семантикой используемых знаний, с организацией решателя задач и пользовательского интерфейса.

В процессе проектирования ИС возникает проблема интеграции ИС и их компонентов. Суть данной проблемы заключается в разработке таких принципов построения ИС, которые обеспечивали бы достаточно простую и автоматизируемую процедуру их интеграции. Решение проблемы интеграции интеллектуальных систем позволит существенно сократить сроки проектирования интеллектуальных систем за счет использования результатов прошлых разработок (как целых ИС, так и различных их фрагментов). При этом типовые фрагменты, которые могут быть многократно использованы в новых проектируемых системах, можно объединить в библиотеки. На использовании таких библиотек и основывается идеология компонентного (модульного) проектирования.

Подход к решению проблемы интеграции интеллектуальной системы состоит в следующем:

- разработка такого способа представления знаний, который обеспечил бы их легкую интегрируемость. В качестве указанного способа предлагается использовать язык унифицированного кодирования семантических сетей;
- трактовка машины обработки знаний (решателя задач) как многоагентной системы, каждый агент которой осуществляет определенное целенаправленное изменение состояния хранимой базы знаний. Очевидно, что добавление в многоагентную систему нового корректно работающего агента не требует внесения каких-либо изменений в другие агенты. В этом заключается важнейшее преимущество многоагентных систем;
- трактовка пользовательского интерфейса интеллектуальной системы как специализированной интеллектуальной системы, имеющей свою базу знаний и свою машину обработки знаний (свой набор агентов).

Очевидно, что интеграция целых интеллектуальных систем предполагает:

- интеграцию баз знаний этих систем;
- интеграцию их машин обработки знаний (решателей задач);
- интеграцию их пользовательских интерфейсов.

В основе предлагаемого подхода к созданию технологии проектирования интеллектуальных систем, направленного на устранение вышеуказанных недостатков, лежат следующие принципы [34]:

1. В качестве формальной основы проектируемых интеллектуальных систем использовать **графодинамические модели** обработки информации, ориентированные на **параллельную и асинхронную** обработку информации.

2. В качестве основы абстрактных логико-семантических моделей интеллектуальных систем, использовать графодинамические модели специального вида – семантические модели представления и обработки знаний, в основе которых лежат **семантические сети**.

3. **Унификация семантического представления знаний** путем определения структуры унифицированных семантических сетей, обеспечивающих представление самых различных видов знаний. Это предполагает разработку соответствующего стандарта, выделяющего из всего многообразия абстрактных языков семантических сетей **базовый универсальный язык семантических сетей** SC-код (Semantic Computercode).

4. **Применение графовых языков программирования**. Для описания способов решения задач и поведения агентов над общей графодинамической памятью необходимо использовать графовые языки программирования, которые ориентированы на обработку унифицированных семантических сетей и программы которых сами являются унифицированными семантическими сетями.

5. **Унификация формального описания агентов**, работающих над семантической памятью. Из всех используемых в интеллектуальной системе графовых языков программирования выделить **базовый графовый язык программирования**, ориентированный на описание агентов, работающих над общей графодинамической памятью, в которой хранятся и обрабатываются унифицированные семантические сети.

6. **Унификация семантических моделей обработки знаний**. На основе унифицированных семантических сетей (sc-текстов) уточнить понятие **унифицированной модели обработки информации**, а также понятие унифицированной модели решения задач. Все указанные абстрактные модели будем называть **sc-моделями обработки знаний** или **sc-машинами**, поскольку в их основе

лежит использование SC-кода. Каждая такая модель (sc-машина) представляет собой многоагентную систему, состоящую:

– из графодинамической памяти, в которой хранятся и обрабатываются тексты SC-кода, – такую память будем называть **sc-памятью**;

– из коллектива агентов, работающих над общей для них sc-памятью и взаимодействующих между собой только через эту память, – такие агенты будем называть **sc-агентами**.

7. **Унификация семантических моделей информационного поиска**. На основе унифицированных семантических сетей (т. е. на основе SC-кода) обеспечить построение **унифицированных семантических моделей информационного поиска** (моделей ассоциативного доступа к информации, хранимой в семантической памяти).

8. **Унификация как семантических моделей интеграции знаний** и моделей интеграции целых интеллектуальных систем, так и **моделей решения задач**.

9. **Унификация визуализации семантических сетей**. В качестве основы организации графического пользовательского интерфейса использовать язык унифицированного визуального представления абстрактных унифицированных семантических сетей в виде, близком к изоморфному. Указанный язык графического изображения sc-текстов назван SCg-кодом (Semantic Codegraphical).

10. **Унификация семантических моделей пользовательских интерфейсов**. Пользовательский интерфейс интеллектуальной системы рассматривать как ИС, предназначенную для **трансляции адресуемых пользователю сообщений** с внутреннего абстрактного семантического языка представления знаний (SC-кода) на тот или иной внешний язык, а также для **трансляции пользовательских сообщений** с внешнего языка на внутренний семантический язык интеллектуальной системы (т. е. в SC-код).

11. В процессе проектирования использовать библиотеку типовых семантически совместимых компонентов ИС и методику модульного проектирования ИС.

12. В рамках данной технологии процесс проектирования ИС должен быть платформенно-независимым.

13. Предлагаемая технология должна быть реализована как **интеллектуальная метасистема**, ориентированная на поддержку проектирования ИС, и построена по тем же самым принципам, что и интеллектуальные системы, разрабатываемые на ее основе.

Указанная интеллектуальная система должна включать в себя:

- теорию (принципы построения) проектируемых интеллектуальных систем, которая входит в состав базы знаний метасистемы;
- библиотеку типовых многократно используемых компонентов (ip-компонентов) интеллектуальных систем, которая входит в состав базы знаний рассматриваемой метасистемы;
- средства автоматизации синтеза, анализа и имитационного моделирования проектируемых интеллектуальных систем и их компонентов (это подсистема интеллектуальной метасистемы, ориентированная на решение задач проектирования интеллектуальных систем);
- интеллектуальную help-систему, являющуюся подсистемой рассматриваемой интеллектуальной метасистемы, ориентированной на информационное обслуживание и обучение разработчиков интеллектуальных систем;
- методику проектирования интеллектуальных систем, которая оформляется как часть базы знаний метасистемы;
- методику обучения проектированию интеллектуальной системы, которая также является частью базы знаний метасистемы;
- интеллектуальную подсистему управления проектированием самой метасистемы;
- интеллектуальную подсистему управления информационной безопасностью метасистемы;
- семейство различных вариантов реализации интерпретаторов унифицированных абстрактных логико-семантических моделей интеллектуальных систем.

Учитывая рассматриваемые выше принципы построения предлагаемой нами технологии, она названа Открытой семантической технологией проектирования интеллектуальных систем (Open Semantic Technology for Intelligent Systems – OSTIS). Соответственно интеллектуальную метасистему, ориентированную на поддержку проектирования интеллектуальных систем, будем называть **метасистемой OSTIS**.

В интеллектуальной метасистеме OSTIS можно выделить целый ряд подсистем, ориентированных на поддержку проектирования различных компонентов интеллектуальных систем, таких, как:

– базы знаний и различные фрагменты баз знаний (онтологии, формальные теории, программы);

– информационно-поисковые машины, машины интеграции знаний, решатели задач;

– пользовательские интерфейсы (графические, естественно-языковые, мультимодальные).

В интеллектуальной метасистеме OSTIS можно также выделить семейство интеллектуальных подсистем, ориентированных на поддержку проектирования различных классов интеллектуальных систем, таких, как:

– интеллектуальные справочные системы (системы информационного обслуживания);

– интеллектуальные обучающие системы (имеющие подсистемы интеллектуального управления обучением);

– интеллектуальные help-системы для пользователей различных компьютерных систем;

– интеллектуальные системы автоматизированного проектирования;

– интеллектуальные системы управления проектами.

4.2. Разработка интеллектуальной справочной системы с использованием технологии OSTIS на примере интеллектуальной геоинформационной системы

Классическая архитектура геоинформационной системы в качестве информационных компонентов включает пространственную и атрибутивную базы данных. Таким образом, все объекты местности имеют пространственную привязку к территории местности, а также задаются характеристики объектов. Для решения прикладных задач требуется установление отношений между объектами местности, и в данном случае, используя инструментальные геоинформационные системы, возможно только установление топологических отношений. Установление других типов отношений, в том числе предметных, весьма затруднительно и может быть решено в частном виде путем создания программ на встраиваемых в инструментальные геоинформационные системы языках

программирования. Причем для установления какого-либо вида семантической связи требуется разработка алгоритма и его программирование.

Создание семантической модели (sc-модели) геоинформационной системы включает в себя создание базы знаний, машины обработки знаний и интеллектуального пользовательского интерфейса со средствами визуального взаимодействия с объектами карты. Особенностью такой модели является представление знаний предметной области в виде семантической сети.

Способ описания различных классов объектов местности. Для каждого объекта местности выделены основные, присущие только ему, семантические характеристики. Особо отметим, что метрические характеристики таким свойством не обладают. Семантические свойства классов объектов местности устанавливаются в соответствии с разработанным и действующим классификатором топографической информации, отображаемой на топографических картах и планах городов ОКРБ 012-2007.

Согласно данному классификатору каждый класс объектов местности имеет уникальное однозначное обозначение. Иерархия классификатора имеет восемь ступеней классификации и состоит из кодов класса, подкласса, группы, подгруппы, отряда, подотряда, вида, подвида. Таким образом, благодаря способу кодирования уже заданы родовидовые связи, отражающие соотношения различных классов объектов местности, а также установлены характеристики конкретного класса объектов местности. В связи с тем, что задаются основные свойства и отношения не конкретных физических объектов, а их классов, то такая информация является по отношению к конкретным объектам местности метаинформацией, а совокупность данной метаинформации представляет собой онтологию объектов местности, которая, в свою очередь, является частью базы знаний интеллектуальной геоинформационной системы.

Онтология объектов местности включает описание следующих классов объектов местности: водные объекты и гидротехнические сооружения; населенные пункты; промышленные, сельскохозяйственные и социально-культурные объекты; дорожная сеть и дорожные сооружения; растительный покров и грунты. Онтология объектов местности представляет собой дерево классификации в соответствии с иерархией, приведенной на рис. 4. Для каждого класса объектов местности установлены родовидовые связи. В качестве примера на рис. 3 приведена иерархия водных объектов.

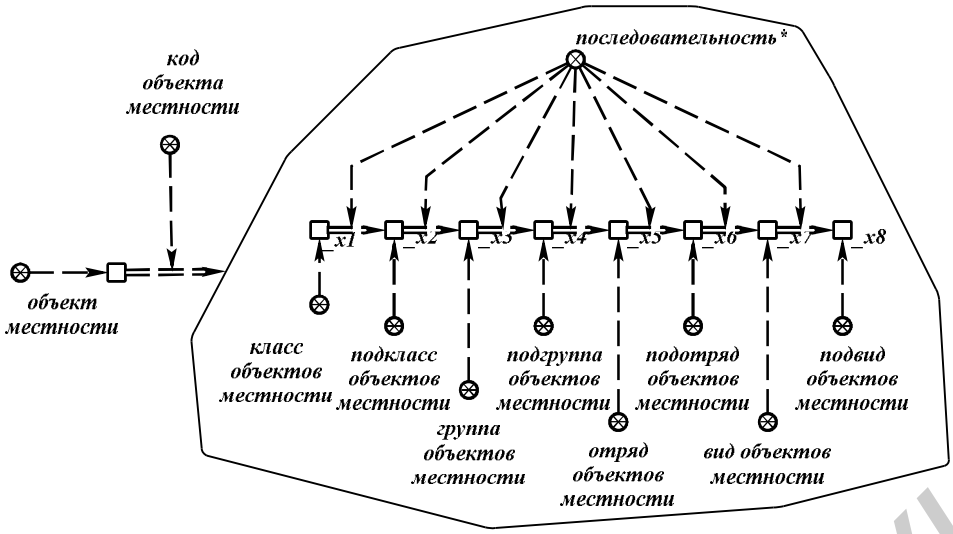

Рис. 3. Уровни иерархии классов объектов местности

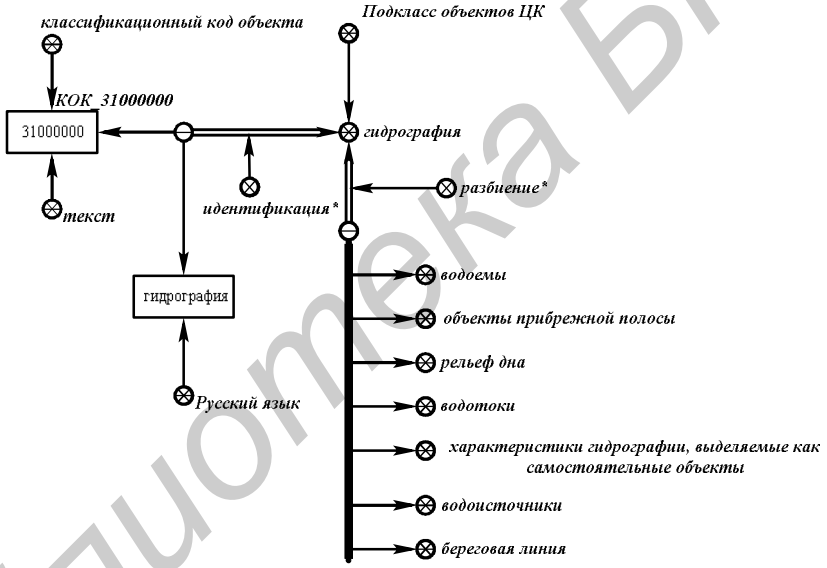

Рис. 4. Иерархия водных объектов местности

Рассмотрим получение онтологии объектов местности и ее представление в sc-модели. На первом этапе создается статья, кодированная с использованием SCn-кода (псевдоестественного языка кодирования семантических сетей). На втором этапе sc.n-статья транслируется во внутреннее представление, т. е. в семантическую сеть специального вида. На рис. 5 в качестве примера проиллюстрировано получение статьи онтологии объектов местности на примере одного из классов объектов местности «реки».

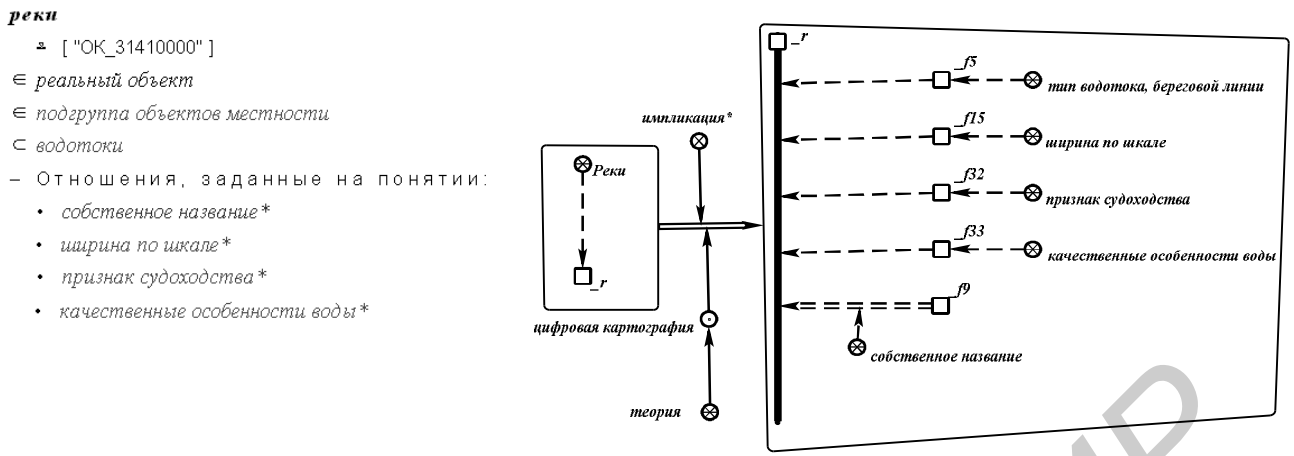

Рис. 5. Статья онтологии объектов местности на примере класса объектов местности «реки»

Кроме статей, описывающих классы объектов местности, в онтологию объектов местности входят статьи с описанием признаков, характеризующих объекты местности. Отметим, что для каждого класса объектов местности выделен свой, характерный только ему, набор признаков (например, для всех объектов местности типа «реки» могут быть заданы отношения «собственное значение*», «ширина по шкале*», «признак судоходства*», «качественные особенности воды*»).

Таким образом, рассмотренная онтология объектов местности и способ ее формального задания позволяют описать все основные классы объектов местности и установить для этих классов набор признаков, характерных для рассматриваемого класса объектов местности, что, в свою очередь, позволяет в дальнейшем создать базу знаний объектов местности с уже установленными родовидовыми отношениями, а также семантическими атрибутами.

Процесс формирования базы знаний. В основу формирования «предметной» базы знаний положен принцип эволюционного проектирования. Это означает, что данный раздел формируется поэтапно, как показано на рис. 6.

Первый этап формирования «предметной» базы знаний – это анализ электронной карты и трансляция в базу знаний объектов местности на заданную территорию. На этом этапе определяется в первую очередь, к какому классу принадлежит исследуемый объект местности, и далее в зависимости от типа объекта формируется статья, соответствующая конкретному физическому объекту местности. Таким образом, создается множество статей, описывающих конкретные объекты местности для каждого класса объектов местности. Следу-

ет отметить, что на данном этапе формирования базы знаний могут быть установлены дополнительные отношения, в частности, отношения принадлежности и топологические отношения.

Второй этап формирования «предметной» базы знаний – это интеграция с внешними базами знаний. На этом этапе помимо географических знаний могут добавляться знания смежных предметных областей, тем самым становится возможным задание межпредметных связей. Наглядным примером служит интеграция с биологическими классификаторами, которые в реализации представляют собой онтологию объектов флоры и фауны. Такая интеграция расширяет функциональные и интеллектуальные возможности прикладной геоинформационной системы. Отметим, что на данном этапе снимается омонимия в названиях географических объектов, принадлежащих классам населенных пунктов за счет использования классификатора СОАТО (система обозначений объектов административно-территориального деления и населенных пунктов).

Рис. 6. Компоненты баз знаний и этапы их получения

Машина обработки знаний интеллектуальных геоинформационных систем. Одним из достоинств интеллектуальных систем, разработанных по технологии OSTIS, является решение предметных задач, когда нет четкой спецификации и алгоритма ее решения. Это достигается с помощью формирования продукций, которые записываются и хранятся также в базе знаний. Необходимо

отметить, что в процессе решения задач генерируются дополнительные знания, необходимые в процессе вывода, которые могут быть сохранены и в дальнейшем использоваться при решении других задач или участвовать в выводе при ответе на поисковые запросы.

Интерфейс пользователя интеллектуальных геоинформационных систем. Результатом запроса пользователя в справочной системе является ответ в виде сформированной sc-конструкции в памяти ИС. В конструкции ответа, помимо информации о свойствах геообъектов, присутствует картографическая информация. Основная задача, решаемая подсистемой визуализации картографической информации, – подключение просмотрщиков и редакторов, которые позволяют визуализировать знания (в частном случае ответов) в удобной для восприятия форме в зависимости от хранимой картографической информации.

На рис. 7 приведена конструкция ответа на запрос пользователя о местонахождении города Минска. В левой части рисунка задается непосредственно инициируемый вопрос, а в правой части к sc-конструкции в качестве ответа присоединяется картографическая информация об объекте местности, имеющаяся в базе знаний системы.

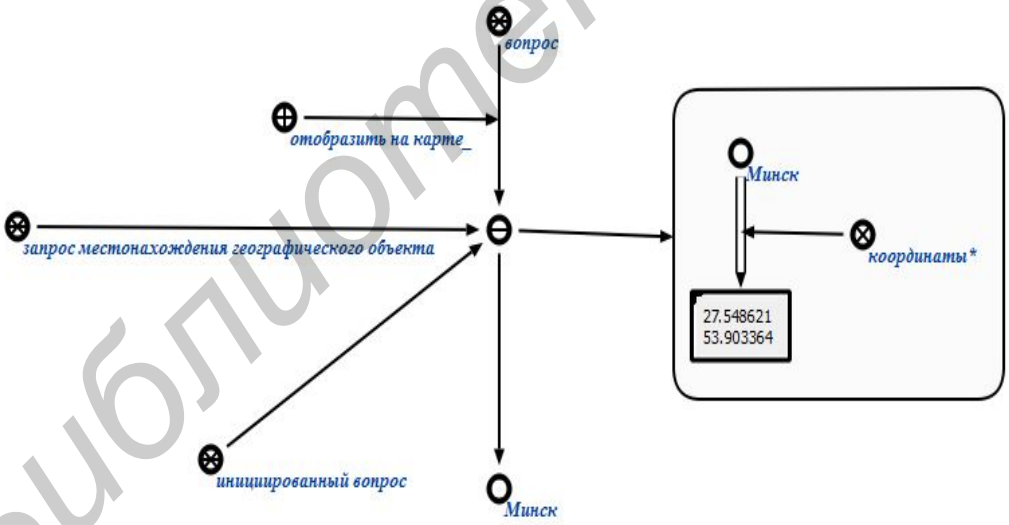

Рис. 7. Представление ответа пользователя в виде семантической конструкции

Следующим этапом является визуализация ответа на карте, интерпретируемой как просмотрщик карт в специализированном sc-окне. С этой целью системой инициируется вызов транслятора, соответствующего формату картографической информации. Например, для картографических данных в системе координат WGS-84 возможно транслирование в формат данных YMapsML

(xml-формат для представления геоданных, разработанный компанией Яндекс). В этом случае для визуализации объектов местности используется картографический сервис http://maps.yandex.ru. Последовательность преобразований представлена на рис. 8.

Шаг 1. Разбор sc-конструкции ответа. На данном шаге выделяется связка отношения *координаты**.

Шаг 2. Определения типа локализации объекта. В результате данного шага определяется один из трех типов локализации: точечный объект, линейный объект, площадной объект в зависимости от количества пар координат в связке отношения *координаты**.

Шаг 3. Выделение названия географического объекта с целью последующего геокодирования и определение, к какому классу объектов местности принадлежит искомый объект.

Шаг 4. Формирование описания отображаемого географического объекта как объекта карты в виде xml-разметки на языке представления геоданных YMapsML. Для рассматриваемого примера будем иметь

```
<ymaps:GeoObject>
        <gml:name>
                Минск
        </gml:name>
        <gml:description>
                город
        </gml:description>
        <gml:Point>
                <gml:pos>
                        27.548621 53.903364
                </gml:pos>
        </gml:Point>
</ymaps:GeoObject>
```

Шаг 5. Формирование уникального имени xml-документа и его отправка на сервер ymapserver.appspot.com под этим именем. Имя сформированного документа сохраняется в содержимом узла sc-окна, которое находится во множестве sc-окон для вывода картографических данных с форматом YMapsML.

Шаг 6. Загрузка с сервера при помощи JavaScript API службы «Яндекс. Карты», сформированного на шаге 4 xml-документа и отрисовка карты.

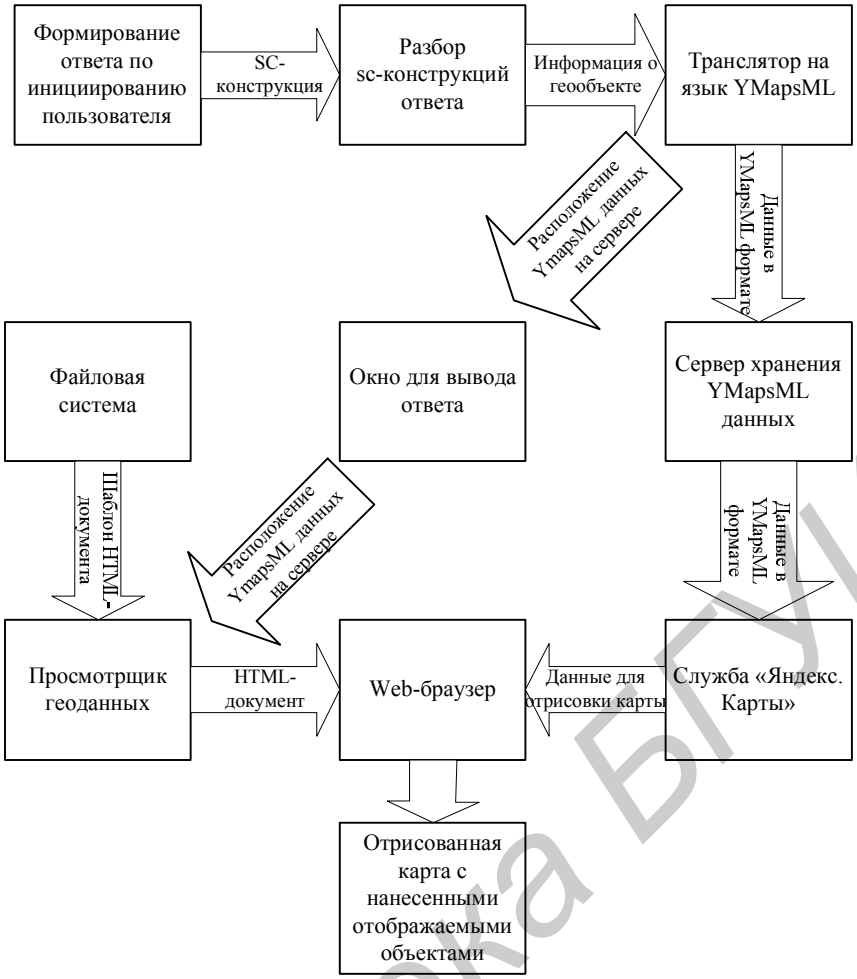

Рис. 8. Последовательность преобразования ответа пользователя в представление на карте

Продемонстрированный подход к проектированию интеллектуальных геоинформационных систем в соответствии с технологией OSTIS позволяет создавать прикладные интеллектуальные системы указанного класса. Причем сроки проектирования сокращаются за счет повторного использования следующих компонентов: онтологии объектов местности, базы знаний объектов местности, машины обработки знаний и картографического интерфейса. В общем случае проектировщику прикладной геоинформационной системы необходимо в соответствии с онтологией объектов местности сформировать базу объектов местности на заданную территорию и записать утверждения предметной области в виде продукций.

5. Архитектура и основные компоненты ПИС и СПР

В настоящее время можно выделить четыре наиболее популярных типа архитектур систем поддержки принятия решений:

1) функциональная система;

2) независимые витрины данных;

3) двухуровневое хранилище данных;

4) трехуровневое хранилище данных.

5.1. Функциональная ПИС

Функциональная ПИС (рис. 9) является наиболее простой с архитектурной точки зрения. Такие системы часто встречаются на практике, в особенности в организациях с невысоким уровнем аналитической культуры и недостаточно развитой информационной инфраструктурой.

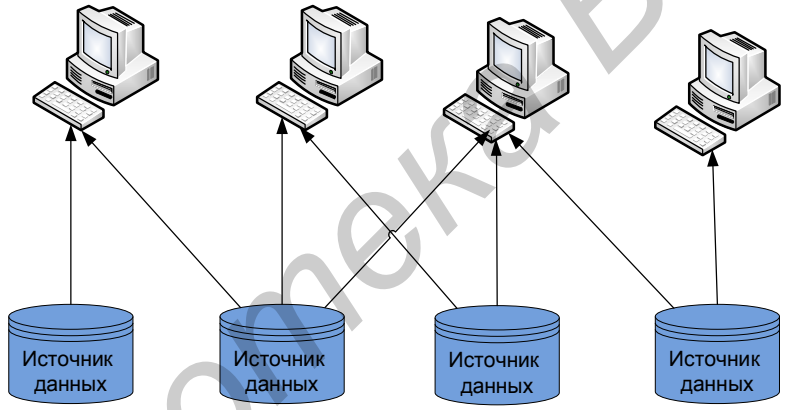

Рис. 9. Функциональная ПИС

Характерной чертой функциональной СПР является то, что анализ осуществляется с использованием данных из оперативных систем. Преимущества СПР с такой структурой следующие:

- быстрое внедрение за счет отсутствия этапа перегрузки данных в специализированную систему;
- минимальные затраты за счет использования одной платформы.

Недостатками подобных систем являются:

- единственный источник данных, сужающий круг вопросов, на которые может ответить система;

- системы характеризуются очень низким качеством данных с точки зрения их роли в поддержке принятия стратегических решений;
- большая нагрузка на систему при выполнении сложных запросов.

5.2. Системы с использованием независимых витрин данных

Независимые витрины данных (рис. 10) часто используются в крупных организациях с большим количеством независимых подразделений, зачастую имеющих свои собственные отделы информационных технологий.

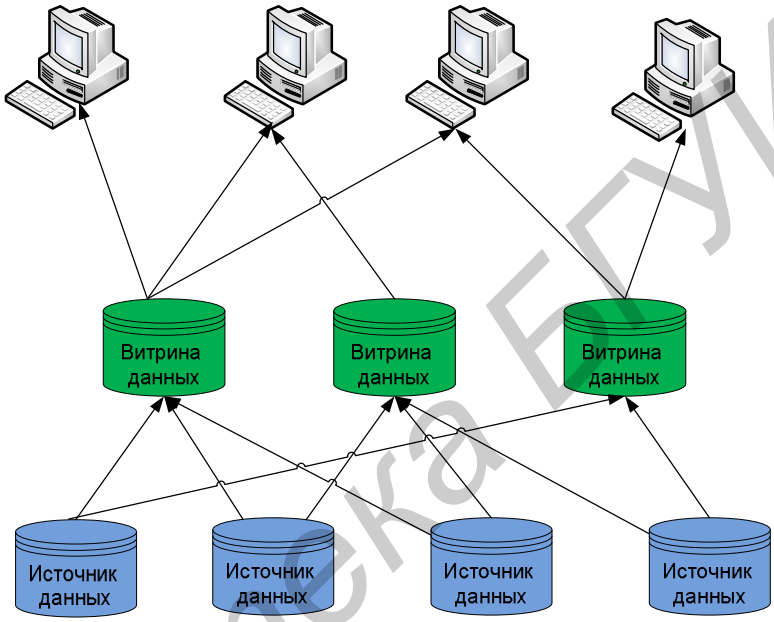

Рис. 10. Независимые витрины данных

Преимущества подобной структуры заключаются в том, что данные в витрине оптимизированы для использования определенными группами пользователей, что облегчает процедуры их наполнения, а также способствует повышению производительности; витрины проектируются для ответов на конкретный ряд вопросов, их можно внедрять достаточно быстро.

Недостатки независимых витрин данных таковы:

- данные хранятся многократно в различных витринах данных, что приводит к их дублированию и, как следствие, к увеличению расходов на хранение и потенциальным проблемам, связанным с необходимостью поддержания непротиворечивости данных;
- потенциально очень сложный процесс наполнения витрин данных при большом количестве источников данных;
- данные не консолидируются на уровне предприятия.

5.3. Системы на основе двухуровневого хранилища данных

Двухуровневое хранилище данных (рис. 11) строится централизованно для предоставления информации в рамках организации. Для поддержки такой архитектуры необходимы специалисты в области хранилищ данных.

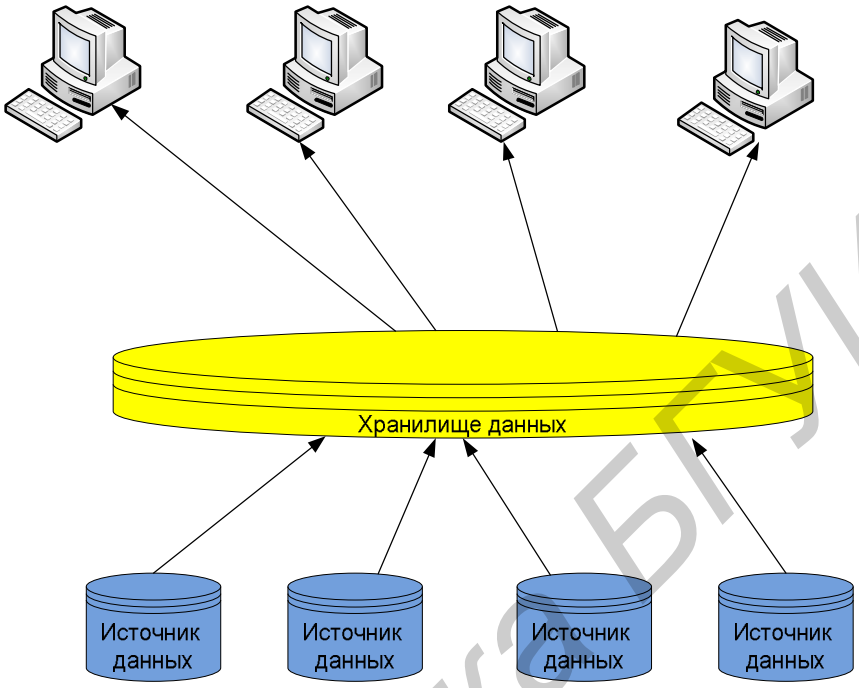

Рис. 11. Двухуровневое хранилище данных

Преимущества двухуровневого хранилища данных:

- данные хранятся в единственном экземпляре;
- затраты на хранение данных минимальны;
- отсутствуют проблемы, связанные с синхронизацией нескольких копий данных.

Недостатки двухуровневого хранилища данных:

- данные не структурируются для поддержки потребностей отдельных пользователей или групп пользователей;
- возможны проблемы с производительностью системы и с разграничением прав пользователей на доступ к данным.

5.4. Системы на основе трехуровневого хранилища данных

Хранилище данных представляет собой единый централизованный источник корпоративной информации (рис. 12). Витрины данных представляют

подмножества данных из хранилища, организованные для решения задач отдельных подразделений компании. Конечные пользователи имеют возможность доступа к детальным данным хранилища, в случае если данных в витрине недостаточно, а также для получения более полной картины состояния бизнеса.

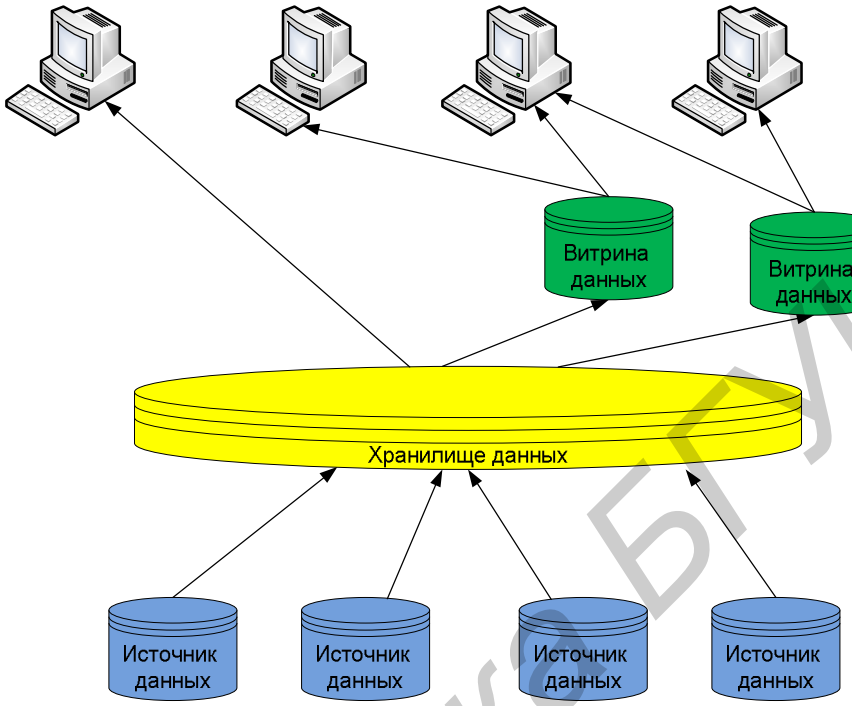

Рис. 12. Трехуровневое хранилище данных

Преимущества такой структуры состоят в следующем:

- создание и наполнение витрин данных упрощено, поскольку наполнение происходит из единого источника очищенных нормализованных данных;
- витрины данных синхронизированы и совместимы с корпоративной моделью данных; существует возможность сравнительно легкого расширения хранилища и добавления новых витрин данных;
- гарантированная производительность.

Недостатки трехуровневого хранилища данных состоят в том, что:

- существует избыточность данных, ведущая к росту требований на хранение данных;
- требуется согласованность с принятой архитектурой.

Прикладные программные системы как традиционные, так и интеллектуальные предназначены в широком смысле для принятия решений. Чтобы процесс принятия решения был более эффективным, создаются специальные системы – системы поддержки и принятия решений. В английском языке исполь-

зуется термин Decision Support Systems (DSS). СПР – это интерактивная автоматизированная система, которая помогает лицу принимающему решение, применять данные и модели для идентификации и решения задач и принятия решений. СПР обладают следующими тремя основными характеристиками:

1. СПР используют и данные, и модели.
2. СПР предназначены для помощи в принятии решений для слабоструктурированных и неструктурированных задач.
3. СПР поддерживают, а не заменяют выработку решений ЛПР.

Классификации СПР. На уровне пользователя СПР делятся на пассивные, активные и кооперативные СПР. Пассивной СПР называется система, которая помогает процессу принятия решения, но не может вынести предложение о решении. Активная СПР может сделать предложение, какое решение следует выбрать. Кооперативная позволяет ЛПР изменять, пополнять или улучшать решения, предлагаемые системой, посылая затем эти изменения в систему для проверки. Система изменяет, пополняет или улучшает эти решения и посылает их опять пользователю. Процесс продолжается до получения согласованного решения.

На концептуальном уровне различаются СПР, управляемые: а) сообщениями (Communication-Driven DSS); б) данными (Data-Driven DSS) СПР; в) документами (Document-Driven DSS); г) знаниями (Knowledge-Driven DSS); д) моделями (Model-Driven DSS).

Управляемая сообщениями СПР поддерживает группу пользователей, работающих над выполнением общей задачи.

СПР, управляемые данными или ориентированные на работу с данными (Data-oriented DSS), также известные как BusinessIntelligence, в основном ориентируются на доступ и манипуляции с данными.

СПР, управляемые документами, управляют, осуществляют поиск и манипулируют неструктурированной информацией, заданной в различных форматах.

СПР, управляемые знаниями, обеспечивают решение задач в виде фактов, правил, процедур.

СПР, управляемые моделями, характеризуются наличием доступа и манипуляциями с математическими моделями (статистическими, детерминированными, оптимизационными, эконометрическими, имитационными и пр.). Не-

которые системы, позволяющие осуществлять сложный анализ данных, могут быть отнесены к гибридным СПР, которые обеспечивают моделирование, поиск и обработку данных.

В зависимости от данных, с которыми эти системы работают, СПР условно можно разделить на *оперативные* и *стратегические*. Оперативные СПР предназначены для немедленного реагирования на изменения текущей ситуации в управлении финансово-хозяйственными процессами компании. Стратегические СПР ориентированы на анализ значительных объемов разнородной информации, собираемой из различных источников. Важнейшей целью этих СПР является поиск наиболее рациональных вариантов развития бизнеса компании с учетом влияния различных факторов, таких как конъюнктура целевых для компании рынков, изменения финансовых рынков и рынков капиталов, изменения в законодательстве и др.

СПР первого типа получили название Информационных систем руководства (Executive Information Systems, ИСР). Они представляют собой конечные наборы отчетов, отражающие в режиме реального времени основные аспекты производственной и финансовой деятельности.

Неотъемлемым компонентом СПР второго типа являются правила принятия решений, которые на основе агрегированных данных дают возможность ЛПР обосновывать свои решения. Стратегические СПР в последнее время активно развиваются. Технологии этого типа строятся на принципах многомерного представления и анализа данных (OLAP).

5.5. Основные компоненты интеллектуальных систем

Интеллектуальные системы предназначены для помощи лицам, принимающим решения при управлении сложными объектами и процессами различной природы, в условиях временных ограничений и наличия различного рода неопределенностей (неполноты, нечеткости и противоречивости исходной информации, недетерминированности стратегии управления и т. д.). Такие системы относятся к классу интегрированных интеллектуальных систем, сочетающих строгие математические методы и модели поиска решения с нестрогими, эвристическими (логико-лингвистическими) моделями и методами, базирующимися на знаниях специалистов-экспертов, моделях человеческих рассуждений и накопленном опыте.

ИС включает ряд взаимодействующих между собой интеллектуальных модулей. К их числу относятся:

- база данных и знаний;
- решатель;
- модуль накопления и пополнения знаний;
- модули объяснения и организации взаимодействия с пользователем;
- модули моделирования проблемной ситуации и прогнозирования;
- модуль связи с внешними объектами;
- модуль организации различных видов интерфейса.

Функциональные возможности перечисленных выше интеллектуальных модулей подробно описаны в [6, 18].

Поиск решения может осуществляться как с использованием традиционных механизмов принятия решений, так и на базе моделей и методов оперирования знаниями.

Структура интеллектуальной системы приведена на рис. 13.

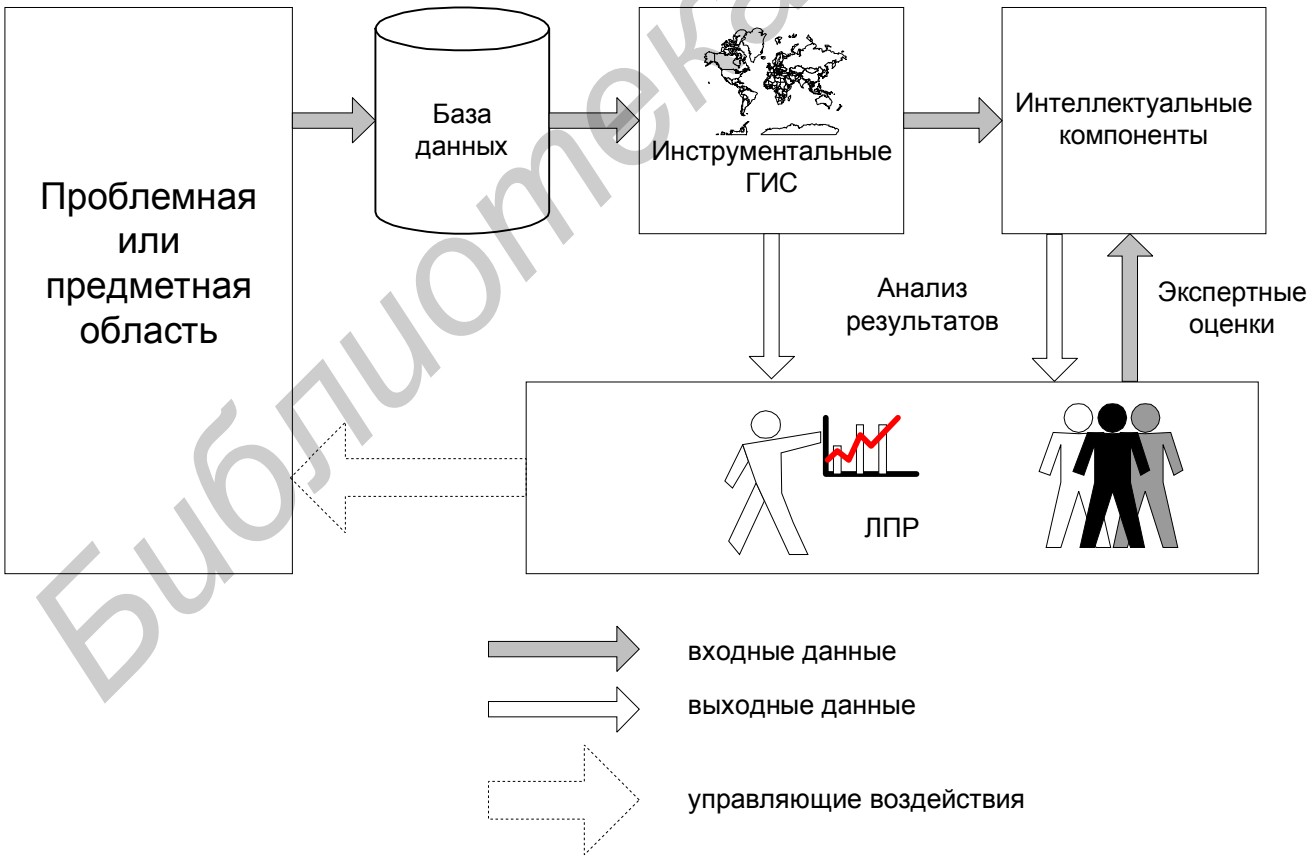

Рис. 13. Обобщенная структура системы, использующей интеллектуальные компоненты

6. Представление знаний в интеллектуальных системах

Системы, основанные на знаниях (СОЗ), оперируют с логическими, объектно-ориентированными и другими моделями, основанными на знаниях экспертов. Вместе с тем СОЗ могут использовать и традиционные алгоритмы.

Системы, основанные на знаниях, представляют собой средства искусственного интеллекта, перспективные для реализации верхних (высокоинтеллектуальных) уровней сложных систем.

6.1. Системы, основанные на правилах

6.1.1. Системы продукционных правил

Под ***системой продукций*** понимается система управления множеством правил, срабатывающих только при выполнении некоторых определенных условий. Условия применимости могут меняться в процессе функционирования продукционной системы в зависимости от получения той или иной информации во время вывода. Условие применимости правила специфицирует некоторые требования к текущему состоянию, а действие содержит описание тех операций, которые надо произвести, если эти требования выполняются.

Применяется и другое определение системы продукций. ***Продукционной системой*** (ПС) называется определенный метод организации вычислительного процесса, при котором программа преобразования некоторой информационной структуры задается в виде множества правил – продукций, каждое из которых представляет собой пару «***условие применимости – действие***». Такому содержательному определению соответствует весьма широкий спектр формальных систем (формальные грамматики, асинхронные программы, потоковые системы). Сюда же можно отнести все формальные системы вывода и доказательств, в которых продукциями являются аксиомы и правила вывода. В ПС обрабатываемая структура обычно содержит информацию о конкретной задаче, а правила представляют знания о предметной области.

Представление продукционных правил. Продукционные правила представляются следующим образом:

<правило>::=(ЕСЛИ<условие>ТО<действие>ИНАЧЕ<действие>

< условие >::=(И{<предложение>}*)

<предложение>::=(ИЛИ{<предложения>}*)|(<предикат><тройка>)

<тройка>::=(<объект><атрибут><значение>)

<действие>::={<заключение>}*|<процедура>

<заключение>::=(<тройка><коэффициент определенности>)

Здесь при истинности условия выполняется действие, стоящее за указателем ТО, а при ложности – действие, стоящее за указателем ИНАЧЕ. Сущности, помеченные звездочкой, могут появиться один или более раз.

Общая постановка задач. Общая постановка задач при использовании ПС формулируется следующим образом. Задаются исходное и целевое состояние задачи. Система на основе заложенных в нее продукционных знаний ищет возможные пути перевода исходного состояния в целевое.

Продукционные системы состоят из трех частей: базы знаний (БЗ), совокупности правил – продукций и интерпретатора. В общем случае база знаний представляет собой ассоциативную память, ориентированную на размещение определенного типа структур. В БЗ реализуются процедуры проверки условий применимости продукций, т. е. процедуры поиска по образцу [6, 18].

Совокупность продукций отражает процедурное содержание знаний, заложенных экспертом данной предметной области в систему. Каждая из продукций имеет вид «условие – действие», где условие – это образец, по которому осуществляется поиск в БЗ, а оператор определяет действия, выполняемые при успешном исходе поиска.

В простейшем случае условия каждой продукции проверяются для всякого текущего состояния БЗ. Из числа продукций, условия которых оказываются выполненными, выбирается одна, для которой выполняется определяемое ею действие. ПС переходит в новое состояние, для которого повторяется та же процедура проверки и выбора. Вычислительный процесс останавливается, когда нет применимых продукций либо когда достигнуто желаемое (целевое) состояние базы знаний. На каждом шаге могут оказаться истинными условия более чем одной продукции. Множество таких продукций обычно называется ***конфликтным***, а процедура выбора продукции из такого множества – ***процедурой разрешения конфликта***. Перечислим некоторые из широко используемых процедур разрешения конфликта:

1) упорядоченные продукции – над продукциями задан полный порядок, и применяется та продукция из конфликтного множества, образцом для которой служит правило с максимальным приоритетом;

2) упорядоченные данные – упорядочиваются элементы базы, и применяется та продукция из конфликтного множества, образцом для условий которой служат данные с максимальным приоритетом;

3) применяется продукция с наиболее жесткими требованиями (правило с самым длинным списком условий);

4) задается иерархия продукций.

Процедуру разрешения конфликта и информацию о последовательности примененных к БЗ правил называют также ***управляющей стратегией***. Выделяют два класса управляющих стратегий: безвозвратную и пробную.

При безвозвратной стратегии в каждый момент вычислений выбирается одно из подходящих правил, оно применяется, и в дальнейшем нет возможности вернуться к этой точке вычислений и применить другое правило, когда текущее вычисление заходит в тупик. При пробной стратегии очередное правило выбирается либо произвольно, либо по категориям, задаваемым процедурой разрешения конфликта. Затем правило применяется, но обеспечивается возможность вернуться позже к этому состоянию базы знаний при вычислении, чтобы применить другое правило.

Интерпретатор можно рассматривать как поисковый процесс, состоящий из двух фаз: выбора правила и его применения.

По способу применения правил различают два типа продукционных систем: прямые и обратные. В ПС, работающих в прямом направлении, образцом для поиска служит левая часть правила (условия). Задача решается в направлении от исходного состояния к целевому. Однако задачу можно решить и в обратном направлении: от целевого состояния БЗ к исходному.

6.1.2. Нечеткие правила

Нечеткие правила также имеют семантику <условие – действие> и являются частным случаем продукционных правил. В нечетких правилах используются нечеткие понятия, а логические заключения делаются с помощью нечеткой логики.

Применение нечетких правил обусловлено тем, что средства искусственного интеллекта позволяют более успешно справиться с неполнотой информации и отсутствием определенности. Методы ИИ дают возможность работать со знаниями, имеющими различную степень неопределенности. Применение не-

четких методов ведет к более высокой степени автоматизации сложных, плохо структурированных процессов. Чтобы реализовать нечеткие методы, необходимо иметь соответствующие знания о процессах, которые могут быть адекватно представлены в терминах нечетких правил.

Существует целый класс описаний, оперирующих качественными характеристиками объектов (*много, мало, сильный, очень сильный* и т. п.). Эти характеристики обычно размыты и не могут быть однозначно интерпретированы, однако содержат важную информацию, например «одним из возможных признаков принадлежности образца к ультраосновным породам является его *повышенная* намагниченность».

Кроме того, в задачах, решаемых интеллектуальными системами, часто приходится пользоваться неточными знаниями, которые не могут быть интерпретированы как полностью истинные или ложные (логические true/false или 0/1). Существуют знания, достоверность которых выражается некоторой промежуточной цифрой, например 0,75.

6.1.3. Системы логического программирования

В системах логического программирования правила также имеют вид импликаций, но в них используются логические модели. Примером систем этого типа является язык Пролог. В логических системах не разрешается использовать отрицание, дизъюнкции и/или равенства. Здесь правила имеют вид «Конъюнкция атомов $\&_i A_i$ влечет атом B»,где атомы могут содержать не только константы, но и переменные.

Импликация $\&_i A_i \rightarrow B$ понимается как процедура, тело которой $\&_i A_i$ состоит из процедурных вызовов A_i. В отличие от продукционных систем и систем нечетких правил в системах логического программирования осуществляется обратный вывод: для атомов, входящих в запрос, отыскивается правило, условие которого следует из этого множества атомов. По этому правилу выводится его заключение, добавляемое затем в виде атома в используемое множество атомов.

Системы логического программирования дают возможность использовать декларативный стиль программирования, когда программисту достаточно в ви-

де логических формул описать предметную область и затем задать вопрос. Интерпретатор Пролога находит ответ на него.

На практике для повышения эффективности поиска вывода используют нелогические процедуры, что позволяет работать не только с задачами логического, но и числового характера. Известны проекты использования систем логического программирования в качестве основы управления в реальном времени с приложениями к созданию полностью автоматических пилотов.

6.2. Системы, основанные на автоматическом доказательстве теорем

Среди систем, основанных на автоматическом доказательстве теорем (АДТ), наиболее часто применяются системы *резолюционного* и *генценовского* типов. В системах первого типа реализуются различные модификации метода резолюций. Этот метод является теоретической основой языков логического программирования типа Пролог. Системы второго типа бывают с естественным выводом (напоминающим манеру рассуждения человека) или с секвенциональным выводом. Здесь вывод представляет многошаговую процедуру декомпозиции (разделения) задач на подзадачи до получения тривиальных подзадач. Системы автоматического доказательства теорем сегодня значительно превосходят другие средства искусственного интеллекта с точки зрения сложности доказываемых ими теорем. Областью применения автоматического доказательства теорем является автоматизация строгих рассуждений, допускающих формализацию, в условиях отсутствия ресурсных ограничений на проведение процедуры доказательства. Это не означает, что методы автоматического доказательства теорем не применимы к задачам, решаемым в реальном времени. Известны примеры использования автоматического доказательства теорем в управлении движущимися объектами для построения полностью автономных систем. АДТ является интеллектуальным уровнем устройства управления, обеспечивающим в режиме реального времени планирование действий, т. е. формирование последовательности команд для достижения поставленной цели.

6.3. Системы, основанные на автоматическом выдвижении гипотез

Интеллектуальные системы обладают важным свойством – способностью извлекать знания, в частности, порождать индуктивное обобщение из имею-

щихся фактов, выдвигать предположения о неизвестных закономерностях или виде некоторой неизвестной функции в частично неопределенной информационной среде. Такие предположения носят название *гипотез*. Автоматическое порождение (выдвижение) гипотез, или автоматическое гипотезирование (АГ), в интеллектуальных системах происходит различными методами. Наибольшую известность приобрели методы обучения: GUHA-метод и ДСМ-метод подробно изложенные в учебном пособии [34].

6.4. Примеры прикладных интеллектуальных систем

Известны случаи применения АДТ в управлении движущимися объектами при построении полностью автоматизированных систем [5]. Примером такого типа является система управления мобильным интегральным роботом STRIPS – самоходным аппаратом, совершающим передвижения в упрощенной среде. Задачей, решаемой STRIPS, является задача «убрать деталь в контейнер». АДТ является интеллектуальным уровнем устройства управления, обеспечивающим в режиме реального времени планирование действий. Этот интеллектуальный уровень более гибко поддерживается моделями в форме логических исчислений, чем жесткими алгоритмами. В модели рассуждений, реализованной в этой системе, используется исчисление предикатов. Для обработки динамически изменяющихся ситуаций вводится время в качестве дополнительного аргумента к тем предикатам, истинность которых меняется в процессе функционирования робота.

Приведем пример использования логического вывода в интеллектуальной системе управления группой лифтов [5]. Группа лифтов – это система кабин, способная перемещаться по этажам, имеющая управляющую систему, которая принимает решения на основе информации о вызовах, положении и маршруте кабин. Простейший алгоритм принятия решения состоит в связывании поступившего вызова с ближайшей в некотором смысле кабиной. Чтобы принять такое решение, необходимо рассмотреть предполагаемое развитие событий в каждом из возможных вариантов и оценить их. В такой системе исключение заведомо худших альтернатив делается на основе логического вывода. Логическая модель группы лифтов есть тройка (F, S, V), где F – формула, описывающая остановку лифта и правила, которым должна подчиняться работа лифта;

S – порядок использования ответов на вопросы при логическом выводе; V – внешние процедуры для имитации работы лифта.

Для проводки грузового судна между островами без вмешательства человека создана экспертная система классических и нечетких правил [5]. Исходными данными являются скорость и угол поворота руля, выходом – курс судна и положение в плоскости (x, y). Нечеткие правила обеспечивают для любого момента времени достаточную близость курса к значению уставки. Уставка и скорость движения, в свою очередь, формируются на основе четких правил. Для перевода судна из одной точки в другую оказывается достаточным нескольких десятков правил, представляющих опыт капитана. Эти правила характеризуют необходимость замедлить судно на поворотах, ускорить на прямых участках и осуществить управление, обеспечивающее судну отслеживание траектории.

Нечеткие правила были использованы в интеллектуальной системе управления длинным многозвенным манипулятором (около 10 м), установленным на космической станции.

В [5] описана бортовая оперативно-советующая экспертная система (ЭС) «Дуэль», предназначенная для интеллектуальной поддержки летчика в типовой боевой ситуации дальнего ракетного боя. Аналогичные системы разрабатываются в Израиле, Великобритании, США. Для описания предметной области использовано пять сценариев, связанных с обеспечением информацией, а также с тактикой ведения боя. ЭС имеет базу знаний интегрированного типа; знания в ней представлены двумя различными способами: продукционным и традиционным алгоритмами, реализующими математические модели. Эти модели предназначены для получения недостающих знаний для ЭС. К ним относятся, например, прогноз текущей ситуации, оценка времени наступления прогнозируемых событий. Продукционные правила служат для формирования рекомендаций и подсказок летчику и содержат объяснения к этим правилам. Правила сгруппированы по сценариям так же, как математические модели. Один сценарий содержит от 20 до 200 правил.

7. Общая характеристика методов принятия решений

Значительная часть проблемно-ориентированных прикладных интеллектуальных систем, предназначенных для интерпретации данных, диагностики, мониторинга, прогнозирования, управления сложными объектами, реализует методы и модели получения информации, необходимой в процессе принятия решения. Кроме того, существуют специальные системы поддержки и принятия решений, направленные на выбор наилучшего варианта действий.

Многие ПИС имеют в своем составе блок математических моделей (оптимального управления, теории игр, статистических решений, машинного обучения и т. д.). Эти модели используются, когда исследуемые варианты действий допускают аналитическое представление или когда часть данных для осуществления формального логического вывода в интеллектуальных компонентах получают классическими математическими методами. Из-за неполноты информации некоторые элементы модели могут оставаться неизвестными. Это происходит, когда сложно найти аналитические законо-мерности, связывающие условия задачи с результатом; отсутствует формализуемая цель управления объектом, отсутствует оптимальность и т. д. В этих случаях для решения задач используют средства ИИ. Поэтому в современных ИС могут совмещаться как традиционные математические методы принятия решений, так и интеллектуальные. В этом разделе будут описаны методы для различных типов проблем теории принятия решений.

7.1. Методы теории принятия решений

Под ***принятием решений*** понимают процесс человеческой деятельности, направленный на выбор наилучшего варианта действий [35]. Модели, описывающие поведение людей, широко используются в исследовании операций. Под ***исследованием операций*** понимают применение математических, количественных методов для обоснования решений во всех областях целенаправленной человеческой деятельности [37].

Под ***операцией*** мы будем понимать систему действий, объединенных единым замыслом и направленных к достижению определенной цели. Операция всегда является управляемым мероприятием. От нас зависит выбор каких-то параметров, характеризующих способ ее организации. Всякий определенный выбор зависящих от нас параметров будем называть ***решением***. Само принятие

решения выходит за рамки исследования операций и относится к компетенции ответственного лица (или группы лиц), которым предоставлено право окончательного выбора.

Варианты действий принято называть *альтернативами*. Для постановки задачи принятия решений необходимо иметь хотя бы две альтернативы. Используя понятие альтернативы, довольно часто процесс принятия решений определяют как обоснованный выбор наилучшей альтернативы из множества альтернатив.

Варианты решений характеризуются различными показателями их привлекательности для ЛПР. Эти показатели называют критериями. ***Критерии оценки альтернатив*** – это показатели их привлекательности для участников процесса выбора, инструмент выбора наилучшей альтернативы. В большинстве задач имеется довольно много критериев оценок вариантов решений. При небольшом количестве критериев (два-три) задача сравнения альтернатив достаточно проста, качества по критериям могут быть сопоставлены. При большом количестве критериев задача усложняется из-за трудностей сопоставления.

Конкретный вид критерия, которым следует пользоваться при численной оценке эффективности той или иной операции, зависит от специфики рассматриваемой операции, а также от задачи исследования и от математической модели, используемой для решения.

Применение тех или иных математических методов обусловлено характером решаемых задач. В науке принятия решений выделяют три типа проблем: хорошо структуризованные, слабоструктуризованные и неструктуризованные проблемы [35]. ***Хорошо структуризованные***, или количественно сформулированные проблемы, – те, в которых существенные зависимости могут иметь численное выражение. ***Слабоструктуризованные***, или смешанные проблемы, – те, которые содержат как качественные, так и количественные элементы, причем качественные, малоизвестные и неопределенные стороны проблем преобладают. Типичные проблемы исследования операций являются хорошо структуризованными. В многокритериальных задачах принятия решений часть информации, необходимой для полного и однозначного решения, отсутствует. Такие проблемы являются слабоструктуризованными.

Существуют проблемы, в которых известен только перечень основных параметров, но количественные связи между ними установить нельзя. В таких

случаях структура, понимаемая как совокупность связей между параметрами, не определена, и проблема называется ***неструктуризованной***.

Для решения хорошо структуризованных задач применяются методы линейного и динамического программирования [36, 37], игровые методы обоснования решений [37], методы теории статистических решений, методы математической статистики и теории вероятностей [38], методы теории массового обслуживания, методы статистического моделирования [37] и т. д. Для решения слабоструктуризованных и неструктуризованных задач используются различные методы оценки многокритериальных альтернатив (экспертные методы, метод анализа иерархий [39], теория полезности [40], теория рисков [40] т. д.), методы искусственного интеллекта, позволяющие моделировать поведение людей при решении тех или иных проблем.

Кроме того, часто методы принятия решений классифицируют не по типам проблем, а в зависимости от условий осуществления выбора. Например, выделяют:

– принятие решений при многих критериях (методы выбора на множестве Парето, методы последовательных уступок, эффективных множеств, метод Электра, анализ иерархий и др.);

– принятие решений в условиях риска (лотереи, теория полезности, теория игорного бизнеса, теория страхования и др.);

– принятие решений в условиях неопределенности (выбор решения как игра с природой, минимаксные и максиминные решения, теория статистических решений и др.). В математической статистике и теории игр состоянием природы принято называть совокупность причин, управляющих ходом случайных событий. Условиями неопределенности считается ситуация, когда результаты принимаемых решений неизвестны либо когда неизвестно, какой из законов случайных событий действует в конкретном случае;

– принятие решений при противодействии (теория игр).

Во всех перечисленных выше условиях правомочно применение методов искусственного интеллекта.

7.2. Модель принятия решений

Рассмотрим математические понятия, с помощью которых будем давать описание процесса принятия решения. Основными среди них являются ***функ-***

ция потерь, ***функция риска***, ***допустимые решающие правила***, ***байесовские*** **и** ***минимаксные*** решающие правила. В теории решений делаются следующие предположения.

Существует ***множество возможных действий*** (вариантов выбора) a, составляющих пространство действий D. Кроме того, существует ***множество состояний*** **(**или ***пространство параметров***) Θ. На пространстве состояний и пространстве действий определена ***функция потерь*** от принятия решения $L(\theta, d)$.

В теории статистических решений вводятся понятия, связанные со случайным характером изучаемых величин. Предположим, что аналитику перед принятием решения разрешено посмотреть на наблюдаемое значение случайной переменной или вектора X, чье распределение зависит от ***истинного состояния природы*** θ (Θ – множество состояний природы, или пространство параметров).

Для каждого $\theta \in \Theta$ существует вероятностная мера P_θ и соответствующая функция распределения $F_x(x|\theta)$, которая представляет собой распределение X при истинном значении параметра. Если X – p-мерный вектор, т. е. $X = (X_1, \ldots, X_p)$, то функция распределения будет функцией p аргументов $F_x(x_1, \ldots, x_p|\theta)$.

Тогда ***задачу статистических решений***, или статистическую игру, можно представить как игру (Θ, D, L), связанную со случайным наблюдением X, распределение которого $F_x(x|\theta)$ зависит от состояния природы θ. В этом описании статистической игры D – множество возможных действий (вариантов выбора), L – потери.

На основании результатов эксперимента $X = x$ (x – наблюдаемое значение X) мы выбираем действие $d(x) \in D$. Такая функция d, которая отображает выборочное пространство на множество D, является ***элементарной стратегией принятия решения, или решающим правилом***.

7.3. Функция потерь и функция риска. Решающее правило

Пусть Θ – множество состояний природы, $\theta \in \Theta$ – некоторое состояние, X – случайная величина описывающая состояние природы.

Пусть за состоянием производятся наблюдения, результат которых выдается в форме прогноза или измерения (решения) $z \in Z$. Z – множество возмож-

ных значений наблюдения. Наблюдение рассматривается как случайная величина Z, распределение которой зависит от состояния природы и описывается семейством условных распределений $P(z|\theta)$, $\theta \in \Theta$ с плотностью распределения (для непрерывного случая) $f(z|\theta)$ или набором вероятностей (для дискретного случая). Все характеристики считаются известными.

Задача заключается в построении решающего правила d: $Z \rightarrow D$, где D – множество возможных решений. Обозначим D_z как множество всех решающих правил (рис. 14). ***Решающее правило*** – это метод, позволяющий выбрать решение, наиболее предпочтительное в каком-либо смысле. *Решающее правило* может быть задано по-разному. Прежде всего, любое решающее правило – это набор условий, которым должны удовлетворять примеры формируемого понятия. Способ задания решающего правила может быть различным (например, логическое выражение, дерево решений, набор продукционных правил, различные математические соотношения). Любая функция $d(x)$, которая отображает выборочное пространство на множество возможных действий A, называется нерандомизированным решающим правилом (РП), или нерандомизированной решающей функцией.

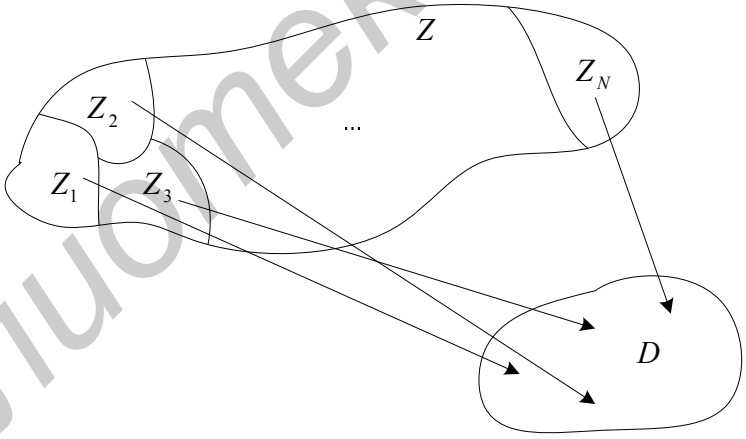

Рис. 14. Множество возможных решений и решающее правило

Построение решающего правила фактически означает разбиение пространства наблюдений Z на непересекающиеся подмножества D_z, такие, что $z \in Z_{d^0} \Rightarrow d = d^0$, где $d^0 \in D$:

$$\bigcup_{d \in D} Z_d = Z,\ Z_{d'} \bigcap Z_{d''} = \varnothing.$$

Риск. Риском (условным риском) от применения решающего правила d при состоянии природы θ будем называть среднее значение потерь:

$R(\theta, d) = \mathrm{E}_{Z|X}(L(\theta, d))$, где L – ограниченная функция потерь;

$R(\theta, d) = \int_Z L(\text{и}, d(z)) f(\text{z}|\theta) dz$ – для непрерывной случайной величины z;

$\mathrm{R}(\theta, d) = \mathrm{E}_{Z|X}(L(\theta, d))\ \mathrm{PR}\{Z = z|\theta\}$ –для дискретной случайной величины z.

Здесь усреднение производится по возможным значениям наблюдений случайной величины Z при дискретномz. Условный риск $R(\theta, d)$ вычисляют до поступления наблюдения z.

Отметим, что если наблюдения вообще не производятся или не несут информации относительно Z, то $d(z) = d = \mathrm{const}(z)$ и $R(\theta, d) = L(\theta, d)$.

Потери от принятия решения $L(\theta, d(x))$ являются случайной величиной. Ожидаемая величина потерь $L(\theta, d(x))$ называется ***функцией риска***

$R(\theta, d) = E_\theta L(\theta, d(x))$

и представляет собой средние потери, когда θистинна и используется $d(x)$. Заметим, что для некоторых видов функции $d(x)$ и значений параметра θ ожидаемое значение потерь может быть равно $\pm\infty$ или даже не существовать.

По аналогии определим условную полезность:

$u(x,d) = E_{Z|\theta}(V(x,d(z))$, где V – ограниченная функция дохода.

7.4. Байесовский и осторожный подходы к построению решающих правил

Осторожный подход к выбору решения. При выборе осторожного решения δ^* минимизируется максимальный риск (минимаксное правило):

$$\max_{\text{и}\in\Theta} R(\theta,\delta^*) = \min_{d\in D_Z} \max_{\text{и}\in\Theta} R(x,d),$$

$$\delta^* = \arg\min_{d\in D_z} (\max_{x\in X} R(x,d))$$

или максимизируется минимальная полезность:

$$\min_{\text{и}\in\Theta} u(\text{и}, \delta^*) = \max_{d\in D_Z} \min_{\text{и}\in\Theta} u(\text{и},d),$$

$$\delta^* = \arg\max_{d\in D_z} (\min_{\text{и}\in\Theta} u(\text{и}, d)).$$

Байесовский подход к выбору решения. При выборе байесовского решения d^* привлекается дополнительная информация о случайной величине x,

которая выражается в виде ее распределения на **Θ**. Как правило, эта информация выражается в виде плотности распределения $p(x)$.

Байесовский (средний) риск от применения решающего правила d равен:

$R(d) = E_{\Theta}(E_{Z|\Theta}(L(\text{и}, d(z)))) = \int\limits_{\text{и}\in\Theta} R(\text{и}, d)\, p(x)dx$ – для непрерывной случай-ной величины X;

$R(d) = E_{\Theta}(E_{Z|\Theta}(L(\text{и}, d(z)))) = \sum\limits_{x\in X} R(x,d)\Pr\{X = x\}$ – для дискретной случайной величины X.

Полезность от применения решающего правила d равна:

$u(d) = E_X(E_{Z|X}(V(x, d(z)))) = \int\limits_{x\in X} V(x, d(z))p(x)dx$ – для непрерывной случайной величины X;

$u(d) = E_X(E_{Z|X}(V(x, d(z)))) = \sum\limits_{x\in X} V(x, d(z))\Pr\{X = x\}$ – для дискретной случайной величины X.

При выборе байесовской решающей функции d^* минимизируется риск:

$$R(d) = E_X(E_{Z|X}(C(x, d(z)))) \to \min_{d\in D_Z},$$

$$d^* = \arg\min_{d\in D_Z} R(d)$$

или максимизируется полезность:

$$u(d) = E_X[E_{Z|X}[V(x, d(z))]] \to \max_{d\in D_Z},$$

$$d^* = \arg\max_{d\in D_Z} u(d).$$

7.5. Допустимые решающие правила

Приведем определения двух правил общего вида, одно из которых использует понятие условного риска, другое – понятие полезности.

Определение 3. Решающее правило d' **доминирует** решающее правило d'', если для любого состояния среды Θ риск, связанный с применением правила d' не превышает риск d'', и, кроме того, существует некоторое состояние окружающей среды, при котором неравенство становится строгим:

$$d' \succ d'' : \begin{cases} \forall x \in X \; R(x,d') \le R(x,d'') \\ u \; \exists x : R(x,d') < R(x,d'') \end{cases},$$

где $d', d'' \in D_Z$.

Определение 4. Решающее правило d' **доминирует** решающее правило d'', если для любого состояния среды X доход, связанный с применением правила d' не меньше дохода d'', и, кроме того, существует некоторое состояние окружающей среды, при котором неравенство становится строгим:

$$d' \succ d'' : \begin{cases} \forall x \in X \; u(x,d') \ge u(x,d'') \\ u \; \exists x : u(x,d') > u(x,d'') \end{cases},$$

где $d', d'' \in D_Z$.

Для каждого из этих видов правил можно ввести определение допустимого правила. Решающее правило d^0 будем называть **допустимым**, если не существует решающее правило, которое бы его доминировало:

d^0 допустимо, если $\nexists d \in D_Z : d \succ d^0$.

Пример 1. На рис. 15 показаны три решающие функции d_1, d_2, d_3. Решающая функция d_3 доминирует решающую функцию: $d_3 \succ d_2$.

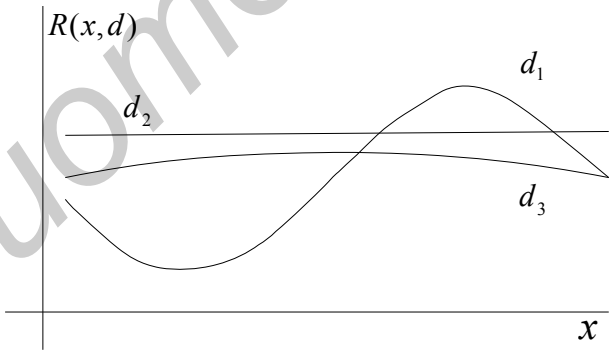

Рис. 15. Наличие доминирующей решающей функции

Пример 2. Решающие функции d_1 и d_2 не следует рассматривать, т. к. они доминируются функциями d_3 и d_4 (рис. 16):

$R(x,d_3) \le R(x,d_1) \; \forall x \in X$ и $\exists x \in X : R(x,d_3) < R(x,d_1)$;

$R(x,d_3) \le R(x,d_2) \; \forall x \in X$ и $\exists x \in X : R(x,d_3) < R(x,d_2)$;

$R(x,d_4) \le R(x,d_1) \; \forall x \in X$ и $\exists x \in X : R(x,d_4) < R(x,d_1)$;

$R(x,d_4) \le R(x,d_2) \; \forall x \in X$ и $\exists x \in X : R(x,d_4) < R(x,d_2)$.

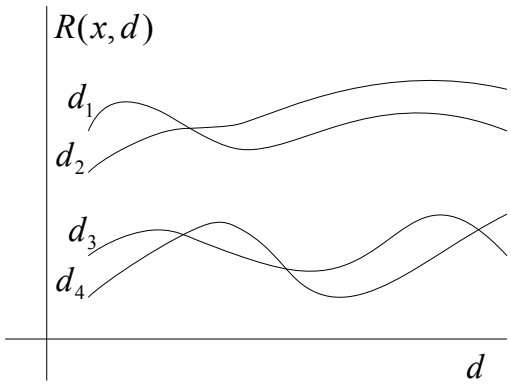

Рис. 16. Выбор решающей функции на основе доминирования

8. Принятие решения при различной исходной информации

8.1. Принятие решений в отсутствие данных и на основе выборочных данных

Наша цель заключается в выборе наилучшего варианта действия с точки зрения потерь. Содержание термина «наилучший вариант» зависит от того, какой информацией о состоянии природы мы располагаем. Наиболее нереалистичной является ситуация, когда параметр θ известен. В этом случае выбор варианта действий прост.

Если θ неизвестен, нам должно быть доступно некоторое априорное распределение $\pi(\theta)$, чтобы осуществить выбор. Помимо этого, нам могут быть известны выборочные данные, на основании которых устанавливается значение θ.

8.2. Принятие решений в отсутствие данных

При отсутствии данных возникают две возможности в зависимости от информации о θ.

1. **Параметр θ известен.** Если θ известно, то выбор наилучшего действия прост. Мы должны выбрать вариант, для которого функция потерь $L(\theta, a)$ так мала, насколько это возможно. Этот случай не представляет интереса для практической деятельности.

2. **Параметр θ неизвестен.** В этом случае мы имеем некоторую информацию о параметре θ в виде априорного распределения $\pi(\theta)$. Эта информация может быть получена на основе прошлого опыта в аналогичных условиях и отражать относительную частоту возникновения различных значений θ. С другой стороны, априорная информация может представлять степень доверия для различных значений θ, основанную на смеси субъективных рассуждений и объективных свидетельств.

Определив $\pi(\theta)$, можно оценить наши возможные действия с помощью априорных ожидаемых потерь:

$$E_\theta L(\theta, a) = \int L(\text{и}, a)\text{р}(\text{и})d\text{и}$$

и в качестве наилучшего варианта действий выбрать тот, который имеет минимальные ожидаемые потери.

8.3. Принятие решений на основе выборочных данных

В ряде случаев мы можем иметь информацию о параметре θ, полученную в виде выборочных данных, полученных в процессе эксперимента, результат которого зависит от величины θ. Обозначим x – результаты наблюдения с функцией правдоподобия $p_\theta(x)$. Снова рассмотрим два случая: 1) параметр θ известен; 2) параметр θ неизвестен.

1. **Параметр θ известен**. Это тривиальный случай, т. к. знание x не добавляет никакой информации, и мы действуем, как в случае отсутствия данных.

2. **Параметр θ неизвестен**. Здесь есть два варианта: 1) отсутствует возможность получить априорную информацию о параметре θ; 2) имеется априорное распределение θ, что увеличивает информацию, получаемую с помощью выборочных данных.

Рассмотрим, каким образом используются сами выборочные данные при выборе действия, связанного с принятием решения. Это в чем-то аналогично использованию выборочных данных при оценивании или проверке гипотез в математической статистике. Там мы рассматриваем отображения выборочного пространства в пространство параметров **Θ**. В теории решений мы устанавливаем области принятия решений в пространстве действий A, а не в пространстве параметров.

Предположим, $d(x)$ устанавливает действие в A, соответствующее имеющимся данным x. Тогда отображение $d(x)$ известно как ***решающее правило***. Оно говорит нам, как действовать, если мы встретились с выборочными данными x. Для каждой конкретной ситуации существует много возможных правил, поэтому необходимо определить, какое из них лучше. Как было указано ранее, оценить решающее правило можно с помощью ожидаемых потерь, т. е. средних потерь по отношению к различным данным, которые могут появиться при принятии решения. Итак, для любого решающего правила $d(X)$ мы должны рассмотреть функцию

$$R(d(X), \theta) = \int_X L(d(x), \text{и}) p_{\text{и}}(x) dx \ .$$

Функция $R(d(X), \theta)$ есть функция θ и известна как ***функция риска***.

Таким образом, любое решающее правило $d(X)$ имеет соответствующую ему функцию риска $R(d(X), \theta)$. Выбор среди нескольких РП производится на основании этой функции риска, при этом выбирается РП с наименьшим значением риска. Однако весьма часто ни одно из правил не имеет наименьший риск одновременно для всех θ. Разработаны многочисленные критерии, позволяющие осуществить выбор $d(X)$, не зная априорного распределения $p_{и}(x)$. Далее в курсе мы будем изучать ***критерии минимкс, макимин, минимин*** и т. д.

Если же априорное распределение $p_{\theta}(x)$ известно, то *наилучшее решающее правило определяется как правило, имеющее минимальный средний риск по отношению к изменениям* θ. Оно известно как **байесовское решающее правило**, которое минимизирует

$$r(d, \pi) = \int_{\Theta} R(d, \text{и})\text{р}(\text{и}) = \int_{\Theta} \text{р}(\text{и})\left(\int_{X} L(d(x)\, p_{и}(x)\right).$$

Получаемый в результате минимальный риск $\min_{d} r(d, \text{р})$ называется байесовским риском.

9. Интеллектуальные методы принятия решений

Взаимное проникновение методов теории принятия решений, искусственного интеллекта и информационных технологий осуществляется в общей области этих дисциплин, называемой интеллектуальными методами принятия решений. Эти методы можно разделить на три группы: первая включает ПР на основе знаний (правила, в том числе нечеткие, логические модели и др.); ко второй относятся методы, используемые в современных информационных технологиях (нейросетевые алгоритмы обработки информации, эволюционные алгоритмы, многоагентные технологии, интеллектуальный анализ данных, машинное обучение и др.); третью группу образуют классические методы ПР, где вычислительные процедуры реализуются не основе интеллектуальных технологий (например, для решения задач многокритериальной оптимизации применяются эволюционные алгоритмы).

Общие цели интеллектуальных методов принятия решений можно сформулировать следующим образом: полнее использовать доступные знания об объекте и среде, чтобы обеспечить решение в соответствии с заданным критерием; искать решение в интеллектуальной манере, прогнозируя изменения в объекте и среде, сохраняя работоспособность при больших изменениях и пересматривая (в случае необходимости) критерии.

В данном разделе будут рассмотрены деревья решений и ассоциативные правила принятия решений. Эти методы предназначены для нахождения закономерностей их данных.

9.1. Деревья решений

9.1.1. Характеристики дерева решений

Дерево решений – это алгоритм классификации, основанный на поиске конъюнктивных закономерностей [38, 41–44]. При построении дерева все конъюнкции строятся одновременно.

Деревом называется конечный связный граф с множеством вершин V, не содержащий циклов и имеющий выделенную вершину $v_0 \in V$, в которую не входит ни одно ребро. Эта вершина называется *корнем дерева*. Вершина, не имеющая выходящих ребер, называется *терминальной* или *листом*. Остальные вершины называются *внутренними*. Дерево называется *бинарным*, если из любой его внутренней вершины выходит ровно два ребра. Выходящие ребра свя-

зывают каждую внутреннюю вершину v с левой дочерней вершиной L_v и с правой дочерней вершиной R_v.

Дерево решений – это дерево, внутренние узлы которого представляют собой проверки для входных наблюдений (примеров, объектов) из обучающей выборки (обучающего множества), а вершины-листы являются категориями, классами (примеров, объектов). Определим требования к данным, с которыми работает алгоритм построения дерева решений:

1. Описание признаков (атрибутов). Вся информация об объектах (примерах) из предметной области должна описываться конечным набором признаков (атрибутов). Каждый признак должен иметь качественное или количественное (числовое) значение. Набор признаков не должен меняться от примера к примеру, и количество признаков должно быть фиксированным для всех примеров.

2. Принадлежность классу. Каждый пример (объект) в обучающей выборке должен быть ассоциирован с конкретным классом, т. е. один из признаков должен быть выбран в качестве имени или номера класса.

3. Объем классов. Классы должны иметь конечное число примеров. Каждый пример должен однозначно относиться к конкретному классу. Количество классов должно быть значительно меньше количества примеров.

Существует несколько характеристик, по которым различаются деревья решений:

1. Проверки могут быть многопризнаковыми (выполняется проверка нескольких признаков входного примера за один раз) или однопризнаковыми (выполняется проверка только одного признака).

2. Проверки могут приводить к двум или более результатам. Если все проверки приводят к двум результатам, то мы получаем двоичное дерево решений.

3. Признаки, которые используются в узлах дерева, могут быть качественными или количественными. Бинарные признаки могут рассматриваться как любые из них.

Классов может быть два или более.

Дерево решений – это способ представления правил в иерархической, последовательной структуре, где каждому объекту соответствует единственный

узел, дающий решение. В результате построения дерева из совокупности исходных данных (множества примеров) извлекаются знания в виде правил.

Под решающим правилом понимается логическая конструкция в форме продукции. Продукционные правила представляются в виде

ЕСЛИ <посылка> ТО <заключение >.

В системах извлечения знаний в качестве посылки выступает описание объекта через его признаки, а заключением будет вывод о принадлежности объекта к определенному классу. В экспертных системах часто используются правила, в которых посылкой является описание ситуации, а заключением – действия, которые необходимо выполнить в данной ситуации.

Любое решающее дерево может быть преобразовано в набор продукционных правил: каждому пути от корня дерева до терминальной вершины соответствует одно продукционное правило. Его посылкой является конъюнкция условий «признак – значение», соответствующих пройденным вершинам и ребрам дерева, а заключением – имя или номер класса, соответствующего терминальной вершине.

Область применения деревьев решений. Область применения деревьев решений в настоящее время широка, но все задачи, решаемые этим аппаратом могут быть объединены в следующие три класса:

1.*Описание данных.* Деревья решений позволяют хранить информацию о данных в компактной форме, вместо них мы можем хранить дерево решений, которое содержит точное описание объектов.

2.*Классификация и принятие решения.* Деревья решений отлично справляются с задачами классификации, т. е. отнесения объектов к одному из заранее известных классов. Целевая переменная должна иметь дискретные значения.

3.*Регрессия.* Если целевая переменная имеет непрерывные значения, деревья решений позволяют установить зависимость целевой переменной от независимых (входных) переменных. Например, к этому классу относятся задачи численного прогнозирования (предсказания значений целевой переменной).

9.1.2. Процедура построения дерева решений

Идею построения деревьев решений на основе примеров рассмотрим по Р. Куинлану (R. Quinlan). Им разработан известный алгоритм ID3 (Induction of Decisiontrees). Пусть задано некоторое обучающее множество *T*, содержащее

объекты (примеры), каждый из которых характеризуется n атрибутами (признаками), причем один из них указывает на принадлежность объекта к определенному классу. Назовем признаки, которые задают свойства каждого примера обучающей выборки, предсказывающими (предикторными) атрибутами. Такие признаки могут быть бинарными, количественными или качественными. Признак, который для каждого примера задает принадлежность к формируемому понятию, называется предсказываемым атрибутом. Этот признак также входит в обучающую выборку.

Пусть через ($i = 1,\ldots,K$) обозначены классы. Тогда существуют три ситуации:

– множество T содержит один или более примеров, относящихся к одному классу C_k. Тогда дерево решений для T – это лист (терминальный узел), определяющий класс C_k;

– множество T не содержит ни одного примера, т. е. пустое множество. Тогда это снова лист, и класс, ассоциированный с листом, выбирается из другого множества отличного от T, скажем, из множества, ассоциированного с родителем;

– множество T содержит примеры, относящиеся к разным классам. В этом случае следует разбить множество T на некоторые подмножества. Для этого выбирается один из признаков, имеющий два и более отличных друг от друга значений O_1, O_2, …, O_s. Множество T разбивается на подмножества T_1, T_2, … T_s, где каждое подмножество T_i содержит все примеры, имеющие значение O_i для выбранного признака. Эта процедура будет рекурсивно продолжаться до тех пор, пока конечное множество не будет состоять из примеров, относящихся к одному и тому же классу.

Вышеописанная процедура лежит в основе многих современных алгоритмов построения деревьев решений. Очевидно, что при использовании данной методики построение дерева решений будет происходить сверху вниз.

Процедура разбиения и правило прекращения разбиения. Для построения дерева на каждом внутреннем узле необходимо найти такое условие (проверку), которое разбивало бы множество, ассоциированное с этим узлом на подмножества. В условие должен быть включен один из атрибутов (признаков). Общее правило для выбора атрибута можно сформулировать следующим образом: выбранный атрибут должен разбить множество так, чтобы получаемые в

итоге подмножества состояли из объектов, принадлежащих к одному классу или были максимально приближены к этому, т. е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше. Такой атрибут считается наиболее информативным среди всех атрибутов, еще не рассмотренных на пути от корня дерева. В качестве меры информативности обычно используются различные критерии, на основании которых производится выбор признака для разбиения множества объектов в узле; ошибка классификации, индекс Джини (Gini), взаимная энтропия.

Рассмотрим узел m, представляющий подмножество R_m, состоящее из N_m объектов. Тогда величина $\hat{p}_{mk} = \sum_{x_i \in R_m} I(y_i = k)$ равна доле наблюдений из класса kв узле m (I – индикаторная функция). Мы относим наблюдение в узле m к классу k, имеющему наибольший объем в узле m: $k(m) = \arg\max_k \hat{p}_{mk}$. Мерами «загрязнения» узла являются:

– ошибка классификации $\frac{1}{N_m} \sum_{i \in R_m} I(y_i = k(m)) = 1 - \hat{p}_{mk(m)}$;

– индекс Джини $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$;

– взаимная энтропия $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$.

Если признаки являются количественными (числовыми), то следует выбрать некий порог, с которым должны сравниваться все значения признака. Пусть количественный признак имеет конечное число значений. Обозначим их $\{v_1, v_2, \ldots, v_n\}$. Предварительно отсортируем все значения. Тогда любое значение, лежащее между v_i и v_{i+1}, делит все примеры на два множества: те, которые лежат слева от этого значения $\{v_1, v_2, \ldots, v_i\}$, и те, что справа $\{v_{i+1}, v_{i+2} \ldots v_n\}$. В качестве порога можно выбрать среднее между значениями v_i и v_{i+1}: $TH_i = \frac{v_i + v_{i+1}}{2}$.

Формулы для оценки «загрязнения» узла последовательно применяются ко всем потенциальным пороговым значениям. Затем среди них выбирается то, которое дает максимальное значение по выбранному критерию. Далее это значение сравнивается со значениями критерия, подсчитанными для остальных

признаков. Если выяснится, что среди всех признаков данный числовой признак имеет максимальное значение по выбранному критерию, то в качестве условия разбиения выбирается именно он.

9.2. Поиск ассоциативных правил

В последнее время неуклонно растет интерес к методам обнаружения или извлечения знаний в базах данных. Выявление закономерностей в больших массивах данных становится основным инструментом для исследования и получения новых знаний в передовых областях науки. Для выполнения автоматизированного анализа больших наборов данных были разработаны специальные методы и алгоритмы, объединенные в рамках направления Data Mining (добыча данных, интеллектуальный анализ данных). Datamining – это процесс обнаружения в сырых данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности [45]. Одним из распространенных методов обнаружения знаний являются алгоритмы поиска ассоциативных правил [43, 46].

Ассоциативные правила позволяют находить закономерности между связанными событиями. Задача поиска ассоциативных правил впервые была представлена для анализа рыночной корзины. Ассоциативные правила эффективно используются в сегментации покупателей по поведению при покупках, анализе предпочтений клиентов, планировании расположения товаров в супермаркетах, маркетинге, адресной рассылке. Их также успешно применяют и в других областях: медицине, для анализа посещений web-страниц (Web Mining), для анализа текста (Text Mining), для анализа данных по переписи населения, в анализе и прогнозировании сбоев телекоммуникационного оборудования и т. д.

Пусть на множестве объектов X задано n бинарных признаков $F = \{f_1, \ldots, f_n\}$, $f_j : X \to \{0, 1\}$ и имеется выборка $X^l = \{x_1, \ldots, x_l\}$ объемом l. Каждому набору признаков φ поставим в соответствие предикат $\varphi(x)$, равный конъюнкции всех признаков из φ:

$$\varphi(x) = \bigwedge_{f \in \varphi} f(x),\ x \in X.$$

Если φ(x) =1, то говорят, что признаки набора φ совместно встречаются у объекта x. Частотой встречаемости или поддержкой (support) набора φ в выборке X^l называется функция

$$\nu(\varphi) = \frac{1}{l}\sum_{i=1}^{l}\varphi(x_i).$$

Набор φ называется часто встречающимся набором (frequent itemset), если его поддержка (support) больше или равна заданному порогу, если $\nu(\varphi) \geq \delta$. Параметр δ называется минимальной поддержкой и в литературе обозначается minSupp.

Пара непересекающихся наборов φ, y называется ассоциативным правилом (association rule) «φ→y», если выполнены два условия:

$$\nu(\varphi \cup y) \geq \delta;$$

$$\nu(\varphi|y) \equiv \frac{\nu(\varphi \cup y)}{\nu(\varphi)} \geq \tau.$$

Величина $\nu(\varphi|y)$ называется значимостью (confidence) правила. Параметр τ называется минимальной значимостью и обозначается minConf. Достоверность правила показывает, в каком проценте случаев из φ следует y.

Задача поиска ассоциативных правил заключается в нахождении всех правил, поддержка и достоверность которых больше, чем некоторый заданный пользователем порог (соответственно minSupp и minConf. В ассоциативном правиле «φ→y» наборы φ и y совместно часто встречаются, и в значительной доле случаев (не менее τ), если встречается набор φ, то встречается также и набор y. Поиск ассоциативных правил принято относить к задачам обучения без преподавателя.

Задача нахождения ассоциативных правил разбивается на две подзадачи:

1. Нахождение всех наборов элементов, которые удовлетворяют порогу minsupport.

2. Генерация правил из наборов элементов, найденных согласно подп.1 с достоверностью, удовлетворяющей порогу minconfidence.

Значения параметров, названных минимальной поддержкой и минимальной достоверностью, выбираются таким образом, чтобы ограничить количество найденных правил. Если величина поддержки большая, то алгоритмы будут находить правила, хорошо известные аналитикам или настолько очевидные, что

нет никакого смысла проводить такой анализ. С другой стороны, низкое значение поддержки ведет к генерации огромного количества правил, что конечно требует существенных вычислительных ресурсов. Тем не менее большинство интересных правил находится именно при низком значении порога поддержки.

Помимо обычного ассоциативного правила существует и обобщенное ассоциативное правило, которое называется импликация $\varphi \rightarrow y$, где ни один из элементов, входящих в набор y, не является предком ни одного элемента, входящего в φ. Поддержка и достоверность подсчитываются так же, как и в случае обычных ассоциативных правил.

Введение дополнительной информации о группировке элементов в виде иерархии дает следующие преимущества:

1. Помогает установить ассоциативные правила не только между отдельными элементами, но и между различными уровнями иерархии (группами).

2. Отдельные элементы могут иметь недостаточную поддержку, но в целом группа может удовлетворять порогу minsupport.

Для нахождения таких правил нужно каждую транзакцию дополнить всеми предками каждого элемента, входящего в транзакцию.

Пример. Задача поиска ассоциативных правил исходно возникла в связи с анализом рыночных корзин. В наиболее распространенном частном случае признаки соответствуют товарам в супермаркете, в общем случае – некоторым предметам. Объекты соответствуют покупкам, точнее чекам или транзакциям. Если $f_j(x_i) = 1$, то в i-м чеке зафиксирована покупка j-го товара. Задача заключается в том, чтобы выявить наборы товаров, которые часто покупают вместе. Например, «если куплен хлеб φ, то будет куплено и молоко y с вероятностью $v(\varphi|y) = 70$ %; причем оба товара покупаются совместно с вероятностью $v(\varphi \cup y) = 5$ %». Величина $v(\varphi|y)$ является оценкой условной вероятности того, что покупатель приобретет набор товаров y, если он уже приобрел набор φ.

Для нахождения ассоциативных правил требуются эффективные алгоритмы, позволяющие решить задачу за приемлемое время. Одним из таких алгоритмов является алгоритм Apriori [47].

Алгоритм Apriori работает в два этапа: на первом шаге необходимо найти часто встречающиеся наборы элементов, а на втором этапе извлечь из них пра-

вила. Количество элементов в наборе будем называть размером набора, а набор, состоящий из k элементов, – k-элементным набором.

На первом шаге алгоритма анализируются 1-элементные часто встречающиеся наборы. Необходимо подсчитать для них поддержку, т. е. частоту их встречаемости в базе данных. Следующие шаги будут состоять из двух частей: генерации потенциально часто встречающихся наборов элементов (их называют кандидатами) и подсчета поддержки для кандидатов.

Недостатком алгоритма Apriori является процесс генерации кандидатов в популярные предметные наборы. Например, если база данных (БД) транзакций содержит 100 предметов, то потребуется сгенерировать 2^{100} кандидатов. Таким образом, вычислительные и временные затраты, которые нужны на их обработку, могут быть неприемлемыми. Кроме этого, алгоритм Apriori требует многократного сканирования базы данных транзакций, а именно столько раз, сколько предметов содержит самый длинный предметный набор. Поэтому был предложен ряд алгоритмов, которые позволяют избежать генерации кандидатов и сократить требуемое число сканирований набора данных.

Эффективной процедурой поиска ассоциативных правил является алгоритм, получивший название *Frequent Pattern-Growth* (**алгоритм FPG**), что можно перевести как «выращивание популярных (часто встречающихся) предметных наборов». Он позволяет не только избежать затратной процедуры генерации кандидатов, но уменьшить необходимое число проходов БД до двух. В основе метода лежит предобработка базы транзакций, в процессе которой эта база данных преобразуется в компактную древовидную структуру, называемую Frequent-Pattern Tree – *дерево популярных предметных наборов*.

Библиографический список

1. Вагин, В. Н. Некоторые базовые принципы построения интеллектуальных систем поддержки принятия решений реального времени / В. Н. Вагин, А. П. Еремеев // Изв. Академии Наук. Теория и системы управления. – 2001. – №6. – С. 114–123.

2. Интеллектуальные технологии в геоинформационных системах : учеб. пособие/ А. Н. Крючков [и др.]. – Минск : БГУИР, 2006. – 201 с.

3. Искусственный интеллект. В 3 кн. Кн. 1 : Справочник. Системы общения и экспертные системы / под ред. Э. В. Попова – М. : Радио и связь, 1990. – 464 с.

4. Железко, Б. А. Теория и практика построения информационно-аналитических систем принятия решений/ Б. А. Железко, А. Н. Морозевич. – Минск : Армита–Маркетинг, Менеджмент, 1999. – 144 с.

5. Интеллектное управление динамическими системами / Б. Е. Федунов [и др.]. – М. : Физматлит, 2002. – 352 с.

6. Статические и динамические экспертные системы : учеб. пособие / Э. В. Попов [и др.]. – М. : Финансы и статистика, 1996. – 211 с.

7. Поспелов, Д. А. Структура исследований в области искусственного интеллекта / Д. А. Поспелов // Лекции всесоюзной школы по основным проблемам искусственного интеллекта и интеллектуальным системам. – Минск, 1990. – Ч. 1. – С. 4–29.

8. Онтологии и тезаурусы: модели, инструменты, приложения / Б. В. Добров [и др.]. – М. : Изд-во ИНТУИТ, 2008. – 173 с.

9. Константинова, Н. С. Онтологии как системы хранения знаний / Н. С. Константинова, О. А. Митрофанова // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы». – 2008. – 54 с.

10. Коваль, С. А. Безэкземплярные и экземплярные онтологии / С. А. Коваль // материалы 36-й Междунар. филолог. конф., Санкт-Петербург, 12 марта 2007 г. [Электронный ресурс]. – Режим доступа : http://www.skoval.ru/research/ontology2007.htm. – Дата доступа : 3.04.2012.

11. Башмаков, А. И. Интеллектуальные информационные технологии / А. И. Башмаков, И. А. Башмаков. – М. : МГТУ им. Н. Э. Баумана. – 2005. – 304 с.

12. Berners-Lee, T. The Semantic Web / T. Berneres-Lee, J. Hendler, O. Lassila // Scientific American Magazine. – 2001. – №284(5). – P. 34–43.

13. Berners-Lee, T. The Semantic Web and Challenges [Electronic resource] / T. Berneres-Lee. – 2012. – Mode of access: http://www.w3.org/2003/Talks/01-sweb-Tbl/overview.html. – Date of access: 27.10.2012.

14. Трофимов, И. В. Эволюция выразительных способностей языка OWL / И. В. Трофимов // Программные системы: теория и приложения : электрон. науч. журн. [Электронный ресурс]. – 2011. – №4(8). – С. 85–94. – Режим доступа : http://psta.psiras.ru/read/psta2011_4_85-94.pdf. – Дата доступа : 18.04.2012.

15. Лапшин, В. А. Система Cyc и ее библиотека онтологий. В 2 ч. / В. А. Лапшин // Искусственный интеллект и принятие решений. – 2010. – №1. – С. 42–53; №2 – С. 40–51.

16. Knowledge Interchange Format [Electronic resource]. – 2010. – Mode of access: http://logic.stanford.edu/kif/dpans.html – Date of access: 18.04.2012.

17. Хорошевский, В. Ф. Пространства знаний в сети Интернет и Semantic Web / В. Ф. Хорошевский // Искусственный интеллект и принятие решений. – 2008. – Ч. 1. – №1. – С. 80–97.

18. WordNet: a lexical database for the English language [Electronic resource]. – 2011. – Mode of access: http://wordnet.princeton.edu. – Date of access: 27.12.2012.

19. Sowa, J. F. Knowledge Representation: Logical, Philosophical, and Computational Foundations / J. F. Sowa. – Boston, MA: PWS Publishing Co.,1995.– 544 p.

20. The project OpenRDF [Electronic resource]. – 2012. – Mode of access: http://www.openRDF.org/doc/papers/Sesame-ISWC2002.pdf. – Date of access: 12.04.2012.

21. Harris, S. SPARQL 1.1 Query Language / S. Harris, A. Seaborne // W3C Working Draft 05 January 2012 [Electronic resource]. – 2012. – Mode of access: http://www.w3.org/TR/sparql11-query/. – Date of access: 12.04.2012.

22. The project HyperGraph DB [Electronic resource]. – 2012. – Mode of access: http://www.hypergraphdb.org. – Date of access: 18.04.2012.

23. TheprojectNeo4j [Electronic resource]. – 2012. – Mode of access: http://dist.neo4j.org/neo-technology-introduction.pdf. – Date of access: 18.04.2012.

24. The project Allegro Graph [Electronic resource]. – 2012. – Mode of access: http://www.franz.com/agraph/allegrograph3.3/. – Date of access: 18.04.2012.

25. The project Open Link Virtuoso [Electronic resource]. – 2012. – Mode of access: http://docs.openlinksw.com/pdf/virtdocs.pdf. – Date of access: 18.04.2012.

26. Alkhateeb, F. Extending SPARQL with regular expression patterns (for querying RDF) / F. Alkhateeb, J.-F. Baget, J. Euzenat // In Web Semantics: science, Services and Agents on the World Wide Web. – 2009. – №7(2). – P. 57–73.

27. Anyanwu, K. SPARQ2L: Towards Support for Subgraph Extraction Queries in RDF Databases / K. Anyanwu, A. Maduko, A. Sheth. // WWW 2007. – 2007. – P. 797–806.

28. Pérez, J. nSPARQL: A Navigational Language for RDF/J. Pérez, M. Arenas, C. Gutierrez // Journal of Web Semantics. – 2010. – №8(4). – P. 255–270.

29. Mulgara-semantic store [Electronic resource]. – 2012. – Mode of access: http://www.mulgara.org. – Date of access: 12.04.2012

30. The Functional Approach to Data Management Modeling, Analyzing and Integrating Heterogeneous Data / P. M. D. Gray [et al.] // LNCS Series, Springer-Verlag. – 2004. – XIII. – 483 p.

31. CubicWeb-The Semantic Web is a construction game! [Electronic resource]. – 2012. – Mode of access: http://www.cubicweb.org. – Date of access: 12.04.2012.

32. Chimezie, O. Versa: Path-BasedRDFQueryLanguage / O. Chimezie // O'RellyXML.com 20 Jule 2005 [Electronic resource]. – 2012. – Mode of access: http://www.xml.com/pub/a/2005/07/20/versa.html?page=1. – Date of access: 27.10.2012.

33. Статистические основы индуктивного вывода : учеб. пособие [доп. МО РБ] / В. В. Голенков [и др.]. – Минск : БГУИР, 2009. – 202 с.

34. Ларичев, О. И. Теория и методы принятия решений, а также хроника событий в волшебных странах : учебник / О. И. Ларичев. – М. : Логос, 2000. – 296 с.

35. Акулич, И. Л. Математическое программирование в примерах и задачах / И. Л. Акулич. – М. : Высш. шк., 1986. – 319 с.

36. Вентцель, Е. С. Исследование операций: задачи, принципы, методология/ Е. С. Вентцель.– М. : Наука, 1988.– 203 с.

37. Вальд, А. Последовательный анализ / А. Вальд. – М. : Физматлит, 1960. – 328 с.

38. Саати, Т. Принятие решений. Метод анализа иерархий / Т. Саати. – М. : Радио и связь, 1993. – 278 с.

39. Райфа, Г. Анализ решений / Г. Райфа – М. : Наука, 1977. – 408 с.

40. Достоверный и правдоподобный вывод в интеллектуальных системах / В. Н. Вагин [и др.]. – М. : Физматлит, 2004. – 712 с.

41. Прикладная статистика: классификация и снижение размерности / С. А. Айвазян [и др.]. – М. : Финансы и статистика, 1989. – 607 с.

42. MachineLearning.ru [Electronic resource]. – 2012. – Mode of access: http://www.machinelearning.ru/wiki/. – Date of access : 19.01.2012.

43. BaseGroup Labs [Electronic resource]. – 1995. – Mode of access: http://www.basegroup.ru/library/analysis/tree/. – Date of access : 3.04.2012.

44. Дюк, В. DataMining: учебныйкурс (+CD) / В. Дюк, А. Самойленко. – СПб. : Питер, 2001. – 368 с.

45. BaseGroup Labs [Electronic resource]. – 1995. – Mode of access: http://www.basegroup.ru/library/analysis/association_rules/. – Date of access: 3.04.2012.

46. BaseGroup Labs [Electronic resource]. – 1995. – Mode of access: http://www.basegroup.ru/library/analysis/association_rules/apriori/]. – Date of access: 3.04.2012.

Св. план 2013, поз. 21

Учебное издание

Самодумкин Сергей Александрович
Степанова Маргарита Дмитриевна
Колб Дмитрий Григорьевич

ПРИКЛАДНЫЕ ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *М. А. Зайцева*
Корректор *Е. И. Герман*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 22.07.2014. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 6,16. Уч.-изд. л. 6,0. Тираж 150 экз. Заказ 151.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6