Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
«Восточно-Сибирский государственный технологический
университет»

Л.В. Найханова

Технология создания методов автоматического построения онтологий с применением генетического и автоматного программирования

Улан-Удэ
Издательство БНЦ СО РАН
2008

УДК 004.02, 004.825, 004.4'24
ББК 32.813
Н208

Научный редактор
Д.-Д.Ш. Ширапов, д-р физ.-мат. наук, проф., заведующий кафедрой «Электронно-вычислительные системы» Восточно-Сибирского государственного технологического университета, г. Улан-Удэ

Рецензенты:
А.Г. Доррер, д-р техн. наук, проф., декан факультета «Автоматизация и информационные технологии» Сибирского государственного технологического университета, г. Красноярск
В.П. Кулагин, д-р техн. наук, проф., заместитель директора ФГУ ГНИИ ИТТ «Информика», г. Москва

Найханова Л.В.
Н208 Технология создания методов автоматического построения онтологий с применением генетического и автоматного программирования: Монография. – Улан-Удэ: Изд-во БНЦ СО РАН, 2008. – 244 с.

ISBN 978-5-7925-0287-1

В монографии рассматриваются вопросы автоматизации процесса создания онтологий. Монография содержит базовые понятия онтологий, категориальный аппарат универсальных онтологий и основные методы их создания. Большое внимание в работе уделено технологиям генетического и автоматного программирования как средствам автоматического построения методов естественно-языковой обработки научного текста, необходимых для создания онтологий.

Книга предназначена для научных и инженерно-технических работников в области информатики и информационных технологий, занимающихся созданием и использованием интеллектуальных систем, а также аспирантов и студентов специальностей в области компьютерных технологий.

УДК 004.02, 004.825, 004.4'24
ББК 32.813

ISBN 978-5-7925-0287-1

Содержание

Введение

Онтологии получили широкое распространение в решении проблем представления знаний и инженерии знаний, семантической интеграции информационных ресурсов, информационного поиска и многих других областей [1, 21, 25, 30, 99, 132, 140].

В области искусственного интеллекта известно следующее определение, данное А.С. Нариньяни [80,81]. Онтология – это комплекс понятий от самых общих до наиболее конкретных, охватывающий полный спектр объектов и отношений, включая события и процессы, а также значения (атрибутов и отношений), определяемые, если необходимо, во времени и пространстве. Эта система сущностей связывается как универсальными зависимостями типа “общее - частное”, “часть - целое”, “причина - следствие” и т.п., так и специфическими для соответствующей модели предметной области. При этом, определяя сущности в онтологии, можно использовать различные аппараты представления знаний, фреймы, слоты которых, например, связываются ограничениями, обуславливающими допустимые сочетания их значений. В качестве ограничений могут выступать продукции, логические, алгебраические, табличные и другие зависимости. Онтология – это модель предметной области, использующая все доступные средства представления знаний, релевантные для данной области. На современном этапе развиваются, в основном, следующие типы онтологий: предметно-ориентированные (Domain-oriented), ориентированные на прикладную задачу (Task-oriented), общие онтологии (Top-level ontologies).

Предметно-ориентированная онтология – это концептуализация мира в понятиях словаря для объектов, их качественных характеристик, отличительных особенностей и т.п. для данной предметной области [109]. Понятия, определенные в словаре, являются принятой в данной предметной области терминологией. Как правило, онтология предметной области содержит простую/сложную таксономию понятий, то есть понятий, наследующих свойства одного (в случае простой таксономии) или нескольких более общих понятий (в случае сложной таксономии). Понятия связаны отношениями с другими основополагающими или определенными в системе понятиями, или неотъемлемыми типами. Онтологии, ограниченные в наследовании свойств различными определениями, включают иерархию понятий, дополнительные отношения (кроме отношения «быть экземпляром»), экземпляры классов и различные виды ограничений (или аксиомы). Отношения обычно имеют несколько собственных свойств, таких, как имя отношения и его описание, закрепленное документально. Аксиомы устанавливают семантические ограничения для системы отношений.

Цель онтологии задач – сделать знания доступными для повторного использования посредством кодирования знаний предметной области на основе стандартного словаря. Онтологии задач определяют степень участия знаний в процессе логического вывода [107].

Данная онтология опирается на предположение о том, что определенные виды распространенных задач могут быть решены с использованием одной и той же модели решения независимо от проблемной области, в которой данные задачи были определены. Абстрактной моделью такого подхода является метод, ориентированный на решение задачи [108] (в дальнейшем просто метод). Методы описывают процесс вывода системы, основанной на знаниях, способом, не зависящим от конкретной реализации и конкретной проблемной области. Окружение метода формируется двумя сторонами: задача, которая должна быть решена, и знания предметной области, которые должны быть применены для ее решения. Таким образом, методы описывают независимые от предметной области компоненты логического вывода, которые устанавливают «закономерности» (например, родовые задачи, схема логического вывода, роли знаний) при использовании знаний предметной области и могут быть применены в различных приложениях, использующих одну и ту же схему вывода [129].

Общая онтология описывает категории – понятия верхнего уровня. Это базовый механизм "разделения мира". Общая онтология связана с понятиями в философском смысле. Примерами могут служить физические, функциональные, поведенческие понятия и отношения, которые относятся к общенаучным понятиям и отношениям.

Ключевым моментом в проектировании онтологий является выбор соответствующего языка спецификации онтологий. Цель таких языков – предоставить возможность указывать дополнительную машинно-интерпретируемую семантику ресурсов, сделать машинное представление данных более похожим на положение вещей в реальном мире, существенно повысить выразительные возможности концептуального моделирования слабоструктурированных Web-данных. Существуют традиционные языки спецификации онтологий (Ontolingua, CycL, языки, основанные на дескриптивных логиках, такие, как LOOM, и языки, основанные на фреймах, – OKBC, OCML, Flogic). Более поздние языки, основанные на Web-стандартах, такие, как XOL, SHOE или UPML, RDF(S), DAML, OIL, OWL, созданы специально для обмена онтологиями через Web. В целом различие между традиционными и Web-ориентированными языками спецификации онтологий заключается в выразительных возможностях описания предметной области и некоторых возможностях механизма логического вывода для этих языков.

В настоящее время разрабатываются онтологии всех трех уровней: верхнего уровня, предметные и задачные [95, 118, 122]. Наиболее известными методологиями создания онтологий являются модель «Аристотель», стандарт онтологического исследования IDEF5, методология METHONTOLOGY, методология Build Domain Ontologies [139]. Модель «Аристотель» предназначена для конструирования онтологии в мультиагентных системах. Стандарт онтологического исследования IDEF5 обладает хорошо проработанной методологией разработки онтологий, имеющей два специализированных языка SL и EL. Однако существует очень

небольшое число программных средств, поддерживающих этот стандарт. Кроме того, он охватывает не все этапы создания онтологии. Для поддержки методологии METHONTOLOGY разработана специальная инструментальная среда, предназначенная для создания онтологии. Методология Build Domain Ontologies имеет наиболее проработанную технологию создания онтологий, которую можно принять за базовый процесс разработки онтологии, добавив первый этап, описанный в работе [95].

Все эти методологии объединяет то, что описание общих понятий, отношений между ними и утверждений при создании онтологии выполняются в интерактивном режиме. Этот процесс является достаточно трудоемким, поэтому не всегда используется. Некоторым решением проблемы может стать построение понятийного базиса онтологии, который впоследствии при необходимости может дополняться или модифицироваться в интерактивном режиме. В связи с этим в работе ставится проблема создания понятийного базиса онтологий на основе автоматического извлечения знаний из источников, имеющих естественно-языковое представление.

Проблема создания понятийного базиса онтологий на основе автоматического извлечения знаний из источников заключается в выборе источников и применении к ним естественно-языковых методов их обработки, позволяющих построить понятийный базис онтологии. К источникам знаний относятся энциклопедии, терминологические и толковые словари, учебники, монографии, научные статьи и т.п., написанные не просто на естественном языке, а на языке науки, основными характеристиками которого являются развернутость, логическая последовательность, связанность, законченность.

Язык науки – это язык повышенной компрессии смысла, в котором существуют предварительное абстрагирование и конденсация мысли, и в котором можно увидеть новые логические связи между единицами знания. Язык науки – это способ объективизации знания. Объективированное научное знание, рассмотренное как семиотическое (знаковое), может исследоваться соответственно в семантическом, синтаксическом и прагматическом аспектах. Язык науки приравнивается либо к системе специальных научных знаний, то есть к самой науке, либо к её теории и логике, в частности, к понятийному аппарату и способам рассуждений и доказательств.

Язык науки, как структурированное научное знание, представляет собой многослойное иерархическое образование, в котором выделяют блоки [82]: терминосистема; номенклатура; средства и правила формирования понятийного аппарата и терминов.

Каждый из этих блоков, в свою очередь, содержит в качестве субэлементов отдельные языковые образования. Так, *номенклатура* включает три уровня: философия, логика и естественный (специализированный) язык; совокупность общенаучных понятий; система конкретно-научных понятий.

Терминосистема состоит из слоев:

1) собственных терминов – эмпирических и теоретических;

2) математических терминов и символов, фиксирующих формальную структуру и отношения абстрактных объектов и понятий науки.

Субэлементами *средств и правил формирования понятийного аппарата и терминов* являются средства доказательства научных теорий и специальные правила образования языковых выражений – грамматика [38].

С другой точки зрения, язык науки состоит из нескольких слоев: объектные и вспомогательные языки; метаязыки. Объектные и вспомогательные языки фиксируют знания об объектах науки. Метаязыки строят и описывают первые.

Объектные языки могут также делиться на несколько слоев. Например, язык современной физики состоит из трех слоев:

– логический, к которому относятся логические правила вывода, выводы, знаки для обозначения кванторов, связок и операций;

– совокупность математических выражений, обслуживающих физические теории;

– собственно физические термины (электрон, масса, сила и т.д.).

В сферу вопросов, связанных с языком науки, входят также вопросы построения определений терминов (дефиниций), описания типов определений и их функционирования в различного вида знаниях, а также вопросы построения, оценки и применения разных типов классификаций – иерархических, фасетных, экстенсиональных и интенсиональных, вопросы различения и классификации синтетических и аналитических суждений, их связи с необходимой и фактуальной истинностью и структурой теоретического знания.

Схематичный процесс отображения знаний из первичных источников в онтологии показан ниже на рисунке В.1.

Энциклопедии, терминологические и толковые словари, на основе которых требуется построить терминосистему предметной области, как правило, имеют более или менее четкую структуру и, как минимум, они состоят из словарных статей. Поэтому необходимо исследовать возможные их структуры с целью распознавания понятий и отношений между ними.

Построение номенклатуры в отличие от терминосистемы несколько сложнее. Если в словарях термины некоторым образом уже выделены, то в научных текстах (учебники, монографии и др.) их необходимо выделять, осуществлять поиск свойств понятий и отношений между понятиями.

Таким образом, для автоматического построения понятийного базиса онтологии, в первую очередь, необходимо определиться с категориями абстракций. Решение этой задачи требует выполнения анализа лингвистических работ в области терминоведения. В результате такой работы будет ясно, какие понятия и отношения необходимо искать в научном тексте и как их классифицировать.

Далее необходимо разработать методы извлечения терминов из источников знаний, их фиксирования и распределения по категориям;

определить способы представления методов. Однако главная проблема заключается в следующем. Извлечение знаний из научных текстов связано с естественно-языковой обработкой информации. Надо отметить, что эти методы постоянно совершенствуются, находятся в постоянном развитии, поэтому их представление должно быть декларативным, так как такое представление обеспечивает наиболее простой способ совершенствования.

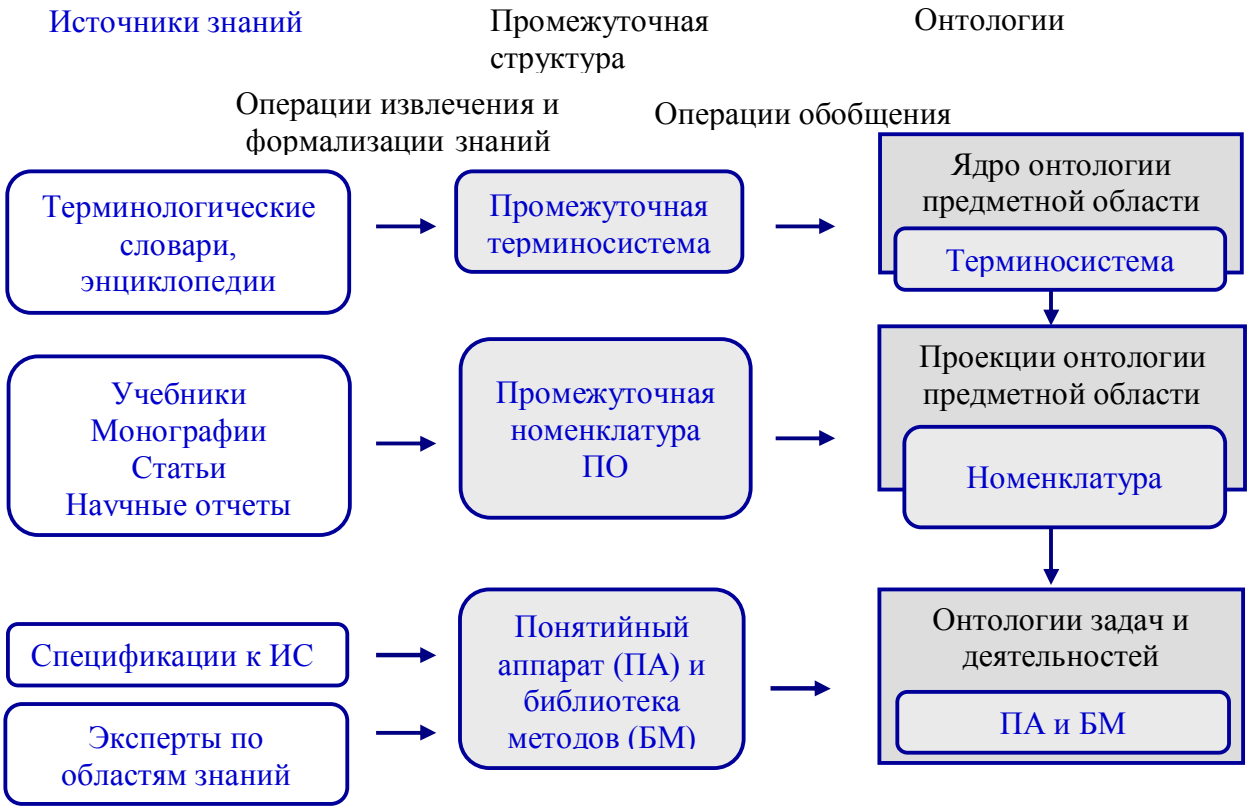


Рисунок В.1 – Отображение знаний из первичных источников в онтологии

Вышеизложенное позволяет сделать вывод о том, что нужна технология, которая позволила бы в почти автоматическом режиме создавать методы естественно-языковой обработки научного текста, которые уже в автоматическом режиме позволили бы строить онтологические модели.

Это означает, что, во-первых, необходимо найти технологию генерации методов на основе знаний о них и, во-вторых, найти технологию автоматического построения кода программного обеспечения.

В работе показано, что в качестве первой можно использовать технологию генетического программирования, которая позволяет генерировать модели решения, а в качестве второй – технологию автоматного программирования, обеспечивающую в автоматическом режиме создание конечных автоматов и преобразователей.

Технологией генетического программирования для генерации автоматов управления системами со сложным поведением занимаются многие ученые различных научных школ в мире [124-126,131]. В России наиболее известной школой, занимающейся такими исследованиями, является Санкт-Петербургская школа по автоматному программированию под руководством профессора А.А.Шалыто. Среди зарубежных

университетов – Массачусетский технологический институт, Университет Южной Калифорнии и другие научные центры, в которых исследования выполняются в следующих прикладных областях: теория игр, робототехника, оптимизация, распознавание, классификация, поиск в заданном пространстве состояний, построение интеллектуальных искусственных систем, построение автоматических преобразователей, проектирование логических схем для автоматов Мили, эволюционное построение клеточных автоматов и т.д. Следует отметить, что в них строятся более простые автоматы, чем те, которые требуются в системах со сложным поведением.

Системы со сложным поведением включают два широких класса систем: реактивные системы и системы с логическим управлением, которые активно разрабатываются на основе технологии автоматного программирования. Результатами разработок являются либо конечные автоматы, либо конечные преобразователи. Исследования данной работы посвящены технологии генетического программирования для генерации конечных автоматов и преобразователей.

Генерация автоматов реактивных систем основана на событийном подходе управления системой. Это означает, что события, возникающие во внешней среде, определяют те или иные действия, которые могут выполняться как в состояниях, так и на переходах. При генерации автоматических автоматов и преобразователей, как систем с логическим управлением, действия, как правило, выполняются только в состояниях, и возникновение того или иного события обуславливает выполнение некоторого действия. Если у автоматов реактивных систем входные переменные и события относятся к входным воздействиям, то у преобразователей события относятся к выходным воздействиям.

Автоматная технология проектирования программных систем [101,102] поддерживается инструментальной системой UniMod [94], которая обеспечивает разработку и выполнение автоматно-ориентированных программ и значительно упрощает разработку автоматов. Но проектирование автоматов, описывающих системы со сложным поведением, является трудоемкой задачей в рамках UniMod, и статическое определение автоматов, то есть полное их задание на этапе написания программы, не обладает достаточной гибкостью. Поэтому необходимо автоматизировать процесс построения модели автомата и отладки его в среде UniMod. Одним из возможных методов проектирования автоматов, соответствующих обозначенным требованиям, являются генетические алгоритмы.

В соответствии с вышеизложенным материал данной работы распределен следующим образом. Первая глава посвящена разработке категориального аппарата онтологий и средствам представления онтологической модели. Во второй главе рассматриваются наиболее интересные методы естественно-языковой обработки научного текста, необходимые для построения онтологий. Третья глава посвящена технологии автоматической генерации методов, рассмотренных во второй главе. В четвертой главе рассматриваются вопросы логического вывода.

Автор сердечно благодарит доцента кафедры систем информатики Н.Б. Хаптахаеву за помощь в подготовке монографии. Выражает большую благодарность доцентам кафедры систем информатики И.С. Евдокимовой, Н.Н. Аюшеевой, С.В. Дамбаевой и С.Д. Даниловой за участие в подготовке данного издания. А также аспиранту Г.А. Хомонову и студентам Б.М. Хандажапову, Н.Г. Кравцову и В.В. Дармахееву за выполнение большого объема вычислительных экспериментов.

Особую признательность автор выражает рецензентам профессору Георгию Алексеевичу Дорреру и профессору Владимиру Петровичу Кулагину за внимательное и в тоже время оперативное ознакомление с рукописью и полезные замечания.

1 Категориальный аппарат построения онтологий

Отправной точкой при создании любой модели знаний о предметной области является выбор ее категориального аппарата. Для любых абстрактных систем, пользующихся одним и тем же тезаурусом или словарем, не существует гарантии, что они смогут правильно использовать одну и ту же информацию, пока не будет принята единая концептуализация. В основе концептуализации лежат категории абстракций, которые связаны с конструкцией термина, лежащего в основе любой онтологии. Поэтому данная глава посвящена обоснованному построению конструкции термина как знака семиотической системы.

1.1 Онтологии и их категориальные аппараты

В настоящее время разработка онтологий – формальных описаний терминов предметной области и отношений между ними – перешла из мира лабораторий по искусственному интеллекту (ИИ) на рабочие столы экспертов по предметным областям. Во всемирной паутине онтологии стали обычным явлением. Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты (сайт Yahoo!), до категоризаций продаваемых товаров и их характеристик (сайт Amazon.com). Консорциумом WWW (W3C) разработан RDF (Resource Description Framework) [105] – язык кодирования знаний на веб-страницах. Он позволяет сделать знания понятными для электронных агентов, которые осуществляют поиск информации. Управлением перспективных исследований и разработок Министерства обороны США (The Defense Advanced Research Projects Agency, DARPA) в сотрудничестве с W3C разработан Язык Разметки для Агентов DARPA (DARPA Agent Markup Language, DAML), являющийся расширением RDF более выразительными конструкциями, предназначенными для облегчения взаимодействия агентов в сети [116]. Во многих дисциплинах сейчас разрабатываются стандартные онтологии, которые могут использоваться экспертами по предметным областям для совместного использования и аннотирования информации в конкретной области. Например, в области химии создана многоуровневая онтология химии [2], в области медицины созданы большие стандартные, структурированные словари, такие, как snomed [137], семантическая сеть Системы Унифицированного Медицинского Языка (the Unified Medical Language System) [117], онтология медицинской диагностики [39]. Также появляются обширные общецелевые онтологии. Например, Программа ООН по развитию (the United Nations Development Program) и компания Dun & Bradstreet объединили усилия для разработки онтологии UNSPSC, которая предоставляет терминологию товаров и услуг (http://www.unspsc.org/).

Онтология определяет общий словарь для ученых, которым нужно совместно использовать информацию в предметной области. Она включает машинно-интерпретируемые формулировки основных понятий предметной области и отношения между ними.

Надо отметить, что к настоящему времени не существует единственного правильного способа моделирования предметной области – всегда существуют жизнеспособные альтернативы. Но необходимо выделить некоторые фундаментальные правила разработки онтологии [133]:

1) лучшее решение почти всегда зависит от предполагаемого приложения и ожидаемых расширений;

2) разработка онтологии – это обязательно итеративный процесс;

3) понятия в онтологии должны быть близки к объектам (физическим или логическим) и отношениям в интересующей предметной области. Скорее всего, это существительные (объекты) или глаголы (отношения) в предложениях, которые описывают избранную предметную область.

В литературе [93,133] онтологии условно делят на специализированные и универсальные. При разработке специализированных онтологий необходимо знание того, для каких целей создается онтология, и насколько детальной или общей она должна быть. Это влияет на многие решения, касающиеся её моделирования. Среди нескольких жизнеспособных альтернатив необходимо определить ту, которая поможет лучше решить поставленную задачу и будет более расширяемой и более простой в обслуживании. После определения начальной версии онтологии необходимо её оценить и отладить, используя ее в приложениях или методах решения задач. В результате почти наверняка нужно будет пересмотреть начальную онтологию. Процесс создания онтологии, являясь процессом итеративного проектирования, может продолжаться в течение всего жизненного цикла онтологии.

Разработка специализированной онтологии начинается с определения ее области и масштаба. Затем определяется функциональность специализированной онтологии, которая должна быть ориентирована на поддержку приложений определенного типа.

С течением времени все более широкое признание получает мысль, что разработка специализированных онтологий, не пригодных для повторного использования (*reusable*), становится непозволительной роскошью. С этой точки зрения основная перспектива видится в движении по направлению к *универсальной* онтологии, которая, по-видимому, должна иметь двухуровневую архитектуру. Верхний уровень образует достаточно детально разработанная общая часть (*Top-Level Ontology*), являющаяся предметом общего соглашения. К ней могут присоединяться доменно и проблемно специализированные концептуальные подсистемы (с возможностью их раздельного администрирования). Движение к *универсальной* онтологии является более предпочтительным, чем разработка и поддержание непростых средств установления соответствий между независимо разрабатываемыми онтологиями. До какого-то момента очевидно, что оба направления могут и должны сосуществовать. Но, как говорит В.Ш.Рубашкин [93], пренебрежение унификацией может, в конце концов, привести к эффекту Вавилонской башни. Его утверждение заключается в том, что только унификация, обеспечиваемая жестко поддерживаемой всеми участниками

движения *Top-Level* онтологией, позволит избежать того печального конца, к которому пришли в свое время попытки разработки интегрированного "многоотраслевого" информационно-поискового тезауруса.

Необходимо помнить, что онтология – это модель реального мира, и понятия в онтологии должны отражать эту реальность.

При разработке *Top-Level* онтологии в первую очередь нужно продумать категориальный аппарат, необходимый для представления знаний о предметной области.

Приведем категории абстракций наиболее известных онтологий. Например, у Sowa J. *независимое понятие* – это понятие, существующее независимо от других объектов; *понятие*, *связанное отношением* – понятие, используемое для определения ролей; *посредствующее понятие* – среда или контекст, где реализуется предыдущее понятие; *непрерывное понятие* и *дискретное понятие*. В системе CYC: *единичный объект* – константа, обозначающая атомарный объект; *неосязаемый объект* – набор нефизических объектов; *временные объекты* – набор понятий, имеющих протяженность. В системе GUM *конфигурация* – это конфигурация элементов, участвующих в некоторой деятельности; *элемент* – единичный самостоятельный объект или понятие; *последовательность* – сложная ситуация, когда различные виды деятельности или конфигурации связаны отношениями таким образом, что они образуют последовательность. В системе WebKB *ситуация* есть объект, «происходящий» во времени и пространстве, *сущность* – объект, который может быть «вовлечен» в ситуацию.

Анализ различных онтологий [31,56,94,95], существующих в настоящее время, показал, что в основном категории абстракций онтологических моделей различны. В связи с этим в работе предлагается категориальный аппарат, выведенный на основе работ лингвистов, логиков и информатиков [3,13,27,45,89].

Определение категориального аппарата связано, с одной стороны, с выявлением концептуальных объектов объективной реальности и отношений между ними, с другой стороны, с их представлением. Действительно одна из интерпретаций языка научных текстов связана с пониманием его как знаковой системы: язык математики, язык химии, то есть с выработанными в разных науках искусственными символическими языками. В них искусственны лексика и синтаксис. Эти языки входят в научный текст, составляя тем самым часть языка науки и делая его естественно-искусственным образованием. Поэтому вначале необходимо определить конструкцию знака категориального аппарата как основы знаковой системы.

1.2 Конструкция знаков

Рассмотрим историю развития представления знаков в семиотике и логике.

1.2.1 Простая схема представления знака

Законы абстрактной семиотики с самого начала её возникновения в виде отдельной науки распределялись по трем ее разделам: синтактика, изучающая отношения между знаками; семантика, изучающая отношения между знаками и обозначаемым предметом; прагматика, изучающая отношения между знаком и человеком.

Общая семиотика рассматривает общие законы, черпая материал для обобщений в разных частных семиотиках. Эти семиотические законы классифицируются в соответствии с названными разделами:

а) объективные законы устройства знаковых систем (синтактика);

б) законы, зависящие от позиции наблюдателя (прагматика);

в) законы смысла (семантика).

Эта классификация, конечно, условна и относительна. Знаковая система всегда понимается как материальный посредник, служащий для обмена информацией между двумя другими материальными системами. Если знаковая система есть материальный посредник между двумя другими материальными системами [96], то такую же природу имеет и знак, который в простейшем случае имеет структуру, показанную на рисунке 1.1.

Рисунок 1.1 – Простейший случай представления знаковой системы

Однако в развитых знаковых системах – языках – знак имеет более сложную конструкцию. Усложнение заключается в том, что те части обеих систем, которые непосредственно контактируют со знаком, в свою очередь, контактируют друг с другом.

1.2.2 Треугольник Фреге

Немецкий логик и математик Готтлоб Фреге [13] впервые ввел определение знака как своеобразное триединство, образованное тремя системами, что показано на рисунке 1.2. Вершина I обозначает предмет, объект, явление действительности, иначе, денотат понятия, вершина II – знак или имя понятия, вершина III – понятие о предмете, объекте, иначе, в лингвистике – десигнат, в математике – смысл имени, или концепт денотата.

Между элементами, обозначенными цифрами I-II-III, имеют место следующие отношения:

1) отношение II-I – отношение от знака к предмету или денотату – называется “иметь обозначение” или “именуется” или “знак имеет предмет”;

2) отношение II-III – отношение от знака к понятию или десигнату – называется “иметь десигнат (смысл)” или “знак выражает смысл” или “знак имеет понятие, смысл”;

3) отношение I-III – отношение от денотата к концепту обозначает, что “концепт денотата определяет денотат” или “предмет имеет понятие, смысл”.

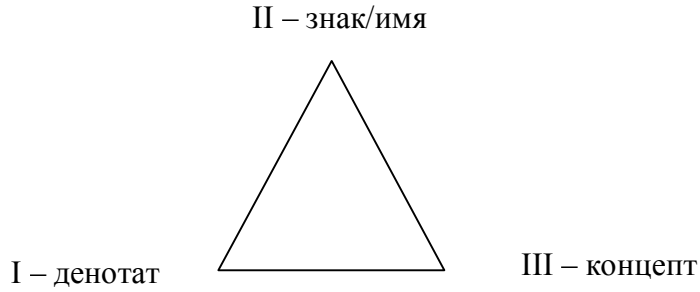


Рисунок 1.2 – Строение знака – треугольник

Такое понимание знака в абстрактной семиотике сложилось к концу XX-го века. Абстрактная семиотика не интересуется какими-либо практическими приложениями в сфере точных наук, техники или искусственного интеллекта. Она остается чисто гуманитарной наукой, основные интересы которой сосредоточены в области культуры, человеческого поведения, искусства и языка.

Прикладная семиотика существенно отличается от абстрактной семиотики. Объектами ее изучения являются не знаки сами по себе, не знаковые системы, а, в первую очередь, применение знаков и знаковых систем в системах представления и использования знаний при решении различных практических задач. На рисунке 1.3 схематически показан треугольник Фреге в интерпретации прикладной семиотики [88]. Сущности реального мира (*денотаты*) вычленяются человеком из общего фона и отражаются в его сознании. В результате отражения возникают представления о денотатах. Представления могут быть образными, акустическими, тактильными или интегральными, объединяющими различные модальности нашего восприятия окружающего мира. Таким образом, *представление* – это тот интегрированный образ (в психологии его часто называют гештальт), с которым имеет дело человек. Денотат, скрывающийся за гештальтом, непосредственно человеку недоступен, а постигается им лишь через некоторые представления, формулируемые его органами чувств или другими источниками информации о реальном мире.

Сталкиваясь с различными денотатами, человек постепенно накапливает о них определенную информацию. Некоторые денотаты кажутся ему неразличимыми, и он воспринимает эти сущности как проявление одной и той же сущности. Другие чем-то отличаются друг от друга, и человек склонен рассматривать их как сущности разного вида. Чтобы реализовать возможность такого различения, он вводит специфические *имена* (знаки), связывая их с представлениями о том или ином виде сущности. При этом один вид сущностей (то есть сущностей, называемых одним и тем же именем) отличается от другого. Так, на основе врожденной для человека процедуры выявления сходства-различия формируется *понятие* (концепт) о сущностях данного типа.

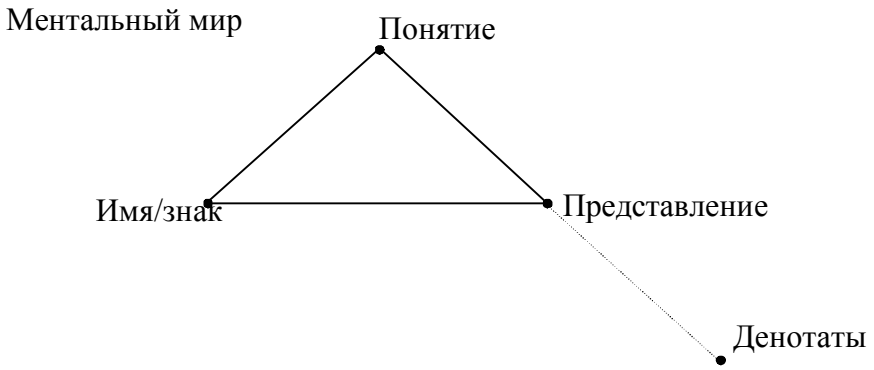


Рисунок 1.3 – Интерпретация треугольника Фреге в прикладной семиотике

В отличие от абстрактной в прикладной семиотике введен новый термин «представление денотата» и дается его определение как интегрированного (унифицированного) денотата сущности. В прикладной семиотике принято следующее определение.

Информационная единица, структурой которой является треугольник Фреге, где вершины отождествляются с именем, понятием и представлением, называется *знаком* или *семой*.

Знак имеет не только видимую внешнюю структуру (рисунок 1.3). То, что соответствует вершинам треугольника, само может быть достаточно сложным образованием. Имя, например, может иметь собственную структуру и нести информацию о специальном виде связей между знаками. В прикладной семиотике их называют *связями наследования*. Отношения наследования образуют иерархическую структуру в системе знаков. Кроме иерархических связей знаки могут объединяться и связями одного уровня. Дальнейшее развитие конструкция знака получила в работах Д.А. Поспелова.

1.2.3 Квадрат Д.А. Поспелова

В прикладной семиотике для поддержки активности иерархических фрагментов сети знаков в семиотическую систему введен метауровень. Для этого предложена конструкция знака, называемая квадратом Д.А.Поспелова [88], показанным на рисунке 1.4. Первая вершина квадрата определяет синтаксис, или способ кодирования знака, вторая – семантику, или понятие о знаке, третья соответствует прагматике – тем процедурам, которые связаны с этим знаком. Четвертая соответствует множеству знаков или фрагменту некоторой структуры на множестве знаков. Она играет роль денотата метазнака и представляет собой фрагмент сети знаков. Данный фрагмент структуры на множестве знаков обладает собственным именем, выделяющим его среди остальных. Это имя представлено в вершине 1. Понятие о фрагменте дано в вершине 2, оно несет в себе информацию о смысле имени, его семантике. Действия, связанные с этим понятием, описаны в вершине 3. Стороны квадрата и его диагональ соответствуют различным процедурам, связывающим компоненты знака. Метазнак образуют вершины 1, 2 и 3 квадрата.

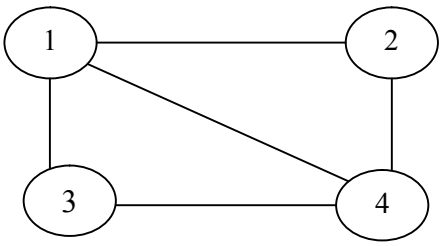


Рисунок 1.4 – Квадрат Поспелова

Таким образом, квадрат Поспелова вводит метауровень в знаковых представлениях. Это позволяет явным образом ввести внутреннюю интерпретируемость действий в знаковые представления, проводить динамическое связывание имен с их денотатами и понятиями. Наличие метауровня также снабжает знаковые системы свойством рефлексии, реализует с помощью семиотических представлений процедуры моделирования и управления, обладающего высокой степенью автономности. Метазнаки в отличие от знаков несут в себе «заряд активности».

Теперь рассмотрим представление термина как знака языка науки, описанное в лингвистических работах [3,27,82].

1.2.4 Пятиугольник С.Е. Никитиной

По С.Е.Никитиной, термин – это знак специальной семиотической системы, обладающий номинативно-дефинитивной функцией. Номинативной – потому, что термин именует, обозначает целый сложный смысловой фрагмент из общей построенной системы интенсионалов (смыслов). Дефинитивной – потому, что замещает дефиницию, состоящую в эксплицитном и/или имплицитном виде из целого ряда высказываний, и подразумевает эту дефиницию в своем употреблении, являясь по отношению к ней вторичным образованием. Как идеальный объект данного рассуждения термин можно назвать именем дефиниции. В деятельностном аспекте – в самом научном творчестве и научной коммуникации – специфика терминологии состоит в осознанности смысла знаков языка науки, то есть в способности говорящего эксплицировать дефиницию употребленного термина. Термины именуют сознательно созданный элемент специального знания.

Вопрос о соотношении значения термина и понятия может решаться по-разному в зависимости от того, что подразумевается под значением и понятием. И.Дальберг предложила схему концепта, показанную на рисунке 1.5. Концепт определяется как единица знания, которая содержит верифицируемые утверждения о выбранном объекте референции и дана в вербальной форме. Такая интерпретация концепта соответствует семантическому треугольнику, в котором признаки (вершина *B*) есть характеристики понятия, представляющие совокупность утверждений о понятии. Если вербальная форма закреплена в научном узусе, то перед нами

термин, значение которого и есть научное понятие, то есть знак или имя сущности. В вершине *A* изображается денотат сущности.

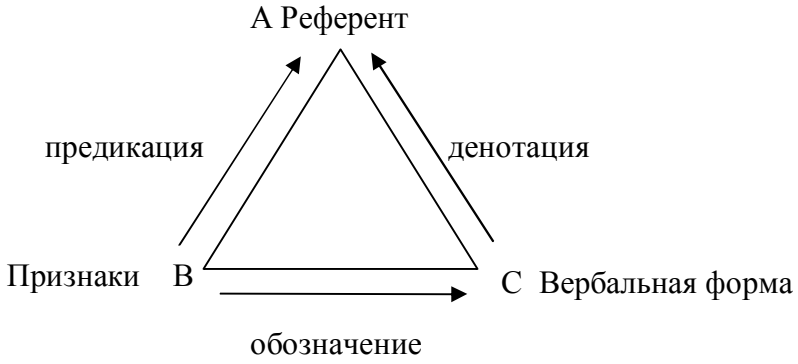


Рисунок 1.5 – Графическая интерпретация концепта по Дальберг

С.Е.Никитиной внесены уточнения, связанные с тем, что между вербальной формой и совокупностью утверждений об объекте, принятой в данном теоретическом языке, стоят ещё два семантических элемента – внутренняя форма и дефиниция. Включив эти два элемента в сферу рассмотрения, ею получен пятиугольник, вид которого показан на рисунке 1.6.

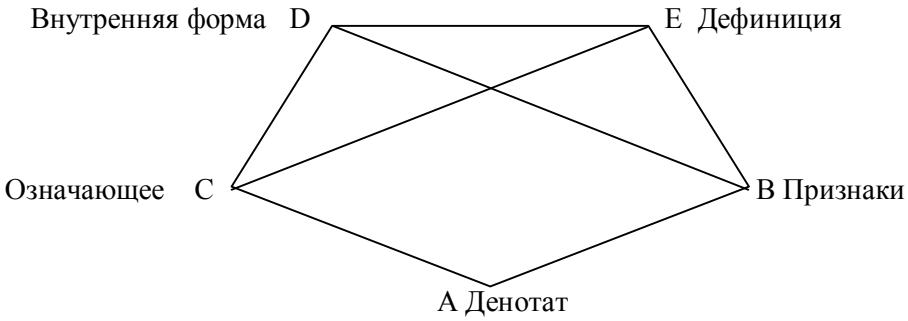


Рисунок 1.6 – Графическая интерпретация концепта

В этой схеме выделены значение (концепт, понятие) и *дефиниция* – словесно выраженный интенсионал, достаточный для задания экстенсионала. Различение имени и внутренней формы необходимо, поскольку не все имена имеют внутреннюю форму. Имена с разной внутренней формой могут иметь общую дефиницию. Например, термины «щелевой» и «фрикативный», указывая на один и тот же денотат, имея одинаковую дефиницию, различаются тем, что во внутренней форме указывают на разные признаки, каждый из которых может быть достаточным для идентификации референта (в данном случае – звука определенного типа).

Внутренняя форма термина (для сложного термина это смысловая структура его частей, составляющих термин) является связующим звеном между логосом и лексисом термина. Она соединяет термин-понятие и термин-слово, оставляя их, тем не менее, достаточно самостоятельными модусами термина. Однако во внутренней форме заключена метафора, конструктивная роль которой в развитии научного знания несомненна. Таким

образом, при анализе понятийного аспекта термина, как правило, невозможно отрешиться от его языковых свойств полностью.

Таким образом, если провести аналогию с треугольником Фреге и квадратом Поспелова, то знак или имя сущности соответствует вершине *C*, в вершинах *B*, *D*, *E* отображается смысл сущности, не имеет соответствия только метазнак. Квадрат Поспелова может описывать любые знаки, но в лингвистических работах конструкция знака более структурирована.

Структурированность знака позволяет более четко описывать понятия и лучше их понимать. Однако структуризация понятия зависит от категорий абстракций.

Таким образом, необходимо определить категории абстракций понятия (термина), чтобы построить более или менее детальную конструкцию знака термина онтологии. Для этого рассмотрим единую интерпретацию метапонятий И.Дальберг [110].

1.2.5 Единая интерпретация метапонятий И.Дальберг

Термин – это знак специальной семиотической системы, являющийся минимальным носителем научного знания, и это краткое название устоявшегося понятия, имеющего дефиницию.

Дефиниция – это такое сочетание форм структурной и субстанциональной дефиниций, при котором из структурной информации вытекает представление о наиболее вероятной субстанциальной, а из субстанциальной информации – наиболее вероятные структурные взаимодействия элементов поля терминосистемы, так что вместе эти два аспекта обеспечивают представление о её целостности и функциональной оправданности. Это означает, что необходимо иметь в качестве субстанциональной дефиниции термина словесное толкование (определение) термина, а в качестве структурной дефиниции – фрагмент сети знаков.

Референт – это представление о денотатах сущности реального мира (объект, явление, процесс), знание о котором описывается в знаковой системе.

Концепт – это знание, которое выражается данным понятием при концептуальном моделировании предметной области.

Интенсионал – это содержание понятия, соответствующее структурной дефиниции и описываемое как внутренняя форма понятия, объединяющая его лексис и логос и достаточная для задания экстенсионала.

Экстенсионал – это объем понятия.

Ниже приведена классификация концептуальных объектов и отношений И. Дальберг. Концептуальные объекты делятся следующим образом:

– *сущность* – материальные и нематериальные объекты, способы их рассмотрения (принципы);

– *свойства* – количественные, качественные, релятивные (отношения);

– *действия* – операции, процессы, состояния;

– *величины* (dimensions) – время, положение, пространство.

Концептуальные отношения делятся следующим образом:

– *квантитативные* (совпадают с теоретико-множественными отношениями тождества, включения, исключения, пересечения, объединения);

– *квалитативные* (в большинстве являются онтологическими и включают в себя иерархические и функциональные отношения).

Эти глобальные классификации представляют собой некоторые априорные схемы научного знания, которые могут накладываться на конкретную терминологию. Классификации показывают, как организуется и воплощается знание в семантической структуре терминологии.

В области искусственного интеллекта, занимающейся моделями представления знаний [36], существует соглашение, что в реальном мире есть объекты. Объекты могут состоять из частей. У объектов есть свойства, которые имеют значения. Объекты могут находиться в различных отношениях друг с другом. Свойства и отношения изменяются во времени. В различные моменты времени возникают события, активизирующие процессы, в которых участвуют объекты и которые изменяются во времени. События могут вызывать другие события, то есть давать эффект. Мир и его объекты могут находиться в различных состояниях.

Таким образом, опираясь на материал, изложенный в данном пункте, построим конструкции термина как знака для сущности (термина-понятия), действия, состояния, события – непременных спутников любого действия, а также свойств и величин.

1.2.6 Конструкция термина как знака

Конструкция знака категории «понятие» построена с учетом представления термина как знака семиотической системы и интерпретации знака С.Е. Никитиной, а также классификации концептуальных объектов И.Дальберг. Вид конструкции знака «Понятие» приведен на рисунке 1.7.

Рисунок 1.7 – Графическая интерпретация знака “Понятие”: *t* – термин; QR_1 – квалитативные отношения; QR_2 – квантитативные отношения, ⊂ – отношение включения

На данном рисунке представлено концептуальное знание о термине. Роль знака играет термин *t*, изображенный на графе в одноименной вершине. Вершина *T* определяет множество терминов *T*, имеющих отношение с термином *t*. Если сравнить данную конструкцию с квадратом Поспелова, то вершине 1 (знак) квадрата соответствует вершина *t*, вершине 2 (смысл знака) – вершины *D*, *C*, *P*, *S* и *T*, вершине 3 (действия, связанные со знаком) – вершина *A*, а вершине 4 (метазнак) – вершина *M*.

Дуги графа соответствуют концептуальным отношениям по классификации И.Дальберга, детализированным на основе работ лингвистов С.Е.Никитиной, Н.Н.Лавровой и других. В связи с тем, что в терминологических словарях, как правило, приводится несколько толкований термина, то, как и в пятиугольнике Никитиной, в конструкцию введена вершина *D*.

Для отражения прагматического аспекта термина в конструкцию знака введена вершина *A*, и так как знания о прагматике несколько отличаются от знаний об объекте, то для них в работе разработаны отдельные конструкции, необходимые для представления знаний о действиях, связанных с объектом.

Таким образом, предложенный подход построения конструкций знака концептуальных объектов Дальберг, как основных категории абстракций, позволяет создать единую концептуализацию предметной области, которую смогут понимать различные системы.

Единая концептуализация необходима для формирования терминосистемы предметной области, являющейся ядром онтологии предметной области. Структура терминосистемы должна определять связи терминов, переходы внутри общей совокупности терминов, описывать семантику, синтактику и прагматику отдельных терминов, а также включать описание набора семантических предикатов (принятые отношения в данной науке), регулярно связывающих термины в научных текстах. Упорядоченная терминосистема может служить одновременно лексикой информационно-поискового языка. С этой позиции она должна представляться в виде тезауруса.

Тезаурус – это максимально полный объем лексики, организованной по тематическому (семантическому) принципу с отражением определенного набора базовых семантических отношений, являющегося полным систематизированным набором данных о какой-либо области знаний, позволяющим человеку или вычислительной машине в ней ориентироваться [1,81]. Данное определение соответствует определению терминосистемы предметной области. Как форма представления терминосистемы тезаурус является хранилищем всех регулярных связей между терминами: парадигматическими и синтагматическими.

Так как модель структуры конкретной науки строится на основе классификаций двух видов: тематической – членение науки на дисциплины, дисциплины – на разделы или направления; и терминологической – категориальной и иерархической, то в тезаурусе должны быть совмещены оба вида классификации. Он может быть устроен как тематический словарь,

но внутри каждого раздела единицы тезауруса связаны между собой иерархическими и неиерархическими отношениями. При таком подходе тезаурус можно рассматривать как модель логико-семантической структуры терминологии, а через неё – как модель структуры соответствующей области науки.

Для построения единого тезауруса в области науки и техники так же, как и для построения отраслевых тезаурусов, важно не только определить лексический состав терминологии, но и используемый набор отношений. Если семантическую структуру науки определить как сеть семантических отношений, соединяющих единицы знания, то семантические отношения, вводимые в тезаурус, должны моделировать эту структуру.

Используя тезаурус как форму представления терминосистемы, являющейся ядром онтологии предметной области, определим структуру словарных статей тезауруса, которую разработаем на основе конструкций знака для выбранных категорий концептуальных объектов [53,62].

1.3 Структуры словарных статей

Представим конструкции знака концептуальных объектов: понятие, действие, состояние, событие, свойство и величины. Эти шесть концептуальных объектов должны быть взаимосвязаны и нести знание о термине-понятии как единицы знания о терминологии предметной области. Конструкция знаков четко определяет структуру словарных статей тезауруса.

1.3.1 Структура словарной статьи «*Понятие*»

В соответствии с конструкцией знака, показанной на рисунке 1.7, словарную статью «Понятие» можно представить в виде восьмерки:

$$Concept = <t, D, P, A, C, S, T, M>, \qquad (1.1)$$

где

t. $t=<t_1, t_2, t_3>$:

t_1 – имя термина;

t_2 – тип концептуального объекта «Понятие»;

t_3 – вид сущности: материальный, нематериальный.

D. $D = \{d_i | \ d_i$ – субстанциальная дефиниция, $i=1 \div n$, n – количество выбранных дефиниций$\}$.

P. $P = \{(p_1, p_2)_i \mid p_1$ – имя свойства; p_2 – ссылка на словарную статью, описывающую p_1; i – количество свойств понятия$\}$.

A. $A = \{(a_1, a_2)_i \mid a_1$ – действие; a_2 – ссылка на словарную статью, описывающую a_1 как термин; $i=1 \div n$, n – количество действий, релевантных термину$\}$.

C. Множество терминов, имеющих квантитативные отношения с термином *t*, описывается двойкой $<C_1, C_2>$, где $C_1 = \{(c_{11}, c_{12})_i \mid c_{11}$ – синоним; c_{12} – ссылка на словарную статью, описывающую c_{11} как термин; $i=1 \div k$, k – количество синонимов$\}$, а $C_2 = \{(c_{21}, c_{22})_i \mid c_{21}$ – коррелят; c_{22} – ссылка на

словарную статью, описывающую c_{21} как термин; i=1÷m, m – количество коррелятов}.

S. S={$(s_1, s_2)_i$ | s_1 – имя термина, описывающего состояние сущности, s_2 – ссылка на словарную статью, описывающую термин s_1, i=1÷n, n=|S|}.

T. Множество понятий (терминов), имеющих квалитативные отношения с термином t, описывается четверкой < T_1, T_2, T_3, T_4>, где

T_1 – понятия, составляющие родовидовые отношения с t, T_1=<T_{11}, T_{12}>, где T_{11} – множество понятий, являющихся родом t; T_{12} – множество понятий, являющихся видом t; элементами множеств T_{11} и T_{12} являются двойки, компоненты двойки – это имя понятия и ссылка на словарную статью, описывающую это понятие;

T_2 – понятия, составляющие отношение «целое–часть» с t; T_2=<T_{21}, T_{22}>, где T_{21} – множество понятий, являющихся целым для t; T_{22} – множество понятий, являющихся частью t; элементами множеств T_{21} и T_{22} являются двойки, компоненты двойки – это имя понятия и ссылка на словарную статью, описывающую это понятие;

T_3 – это двойка T_3=<t_{31}, t_{32}>, где t_{31} – термин, обозначающий способ представления термина t, t_{32} – ссылка на словарную статью, описывающую t_{31} как термин;

T_4 – это двойка T_4=<t_{41}, t_{42}>, где t_{41} – термин, обозначающий способ выражения термина t, t_{42} – ссылка на словарную статью, описывающую t_{41} как термин.

M. С помощью данного элемента определяются отношения между знаковыми системами. Он описывается двойкой <M_1, M_2>, где

M_1 – способ метаязыкового представления, который позволяет зафиксировать связь термина и его представления в метаязыке;

M_2 – термин другого языка, который позволяет зафиксировать связь терминов различных предметных областей.

Каждый элемент M_1 и M_2 также состоит из двух компонентов, первый из которых является именем термина (способа метаязыкового представления или термина другого языка), а второй – ссылкой на словарную статью, в которой описан данный термин.

Словарная статья «Понятие» является центральной. При описании термина посредством элементов *P*, *A* и *S* устанавливается связь со словарными статьями, описывающими свойства, действия и состояния. Через элементы *C* и *T* устанавливаются различные связи с другими терминами. Элемент *M* позволяет установить отношения с терминами смежной предметной области.

1.3.2 Структура словарной статьи «*Действие*»

Известны две распространенные в лингвистике модели действия [3,15,19]. Первая – каузальная, в которой действие представляется как причинение, когда микрособытие, происходящее с агенсом (объектом действия), интерпретируется как причина микрособытия, происходящего с

пациенсом (объектом действия). Вторая модель – пропозитивная, в которой действие рассматривается как целостное событие, центр которого образует предикат, окруженный актантной рамкой и сирконстантами. В действии в такой модели участвуют такие актанты, как агенс/агентив – активный исполнитель действия, пациенс или объект/объектив – лицо или предмет, изменяющиеся в результате действия, инструмент – предмет, опосредующий воздействие агенса на объект/пациенса и некоторые другие, определяемые конкретным характером действия.

Обе модели обладают достоинствами и недостатками. Каузальная модель вскрывает причинно-следственную природу действия, однако разрывает несомненно наличествующую в сознании носителя языка связь «события субъекта ↔ события объекта». Пропозитивная модель характеризует действие как целостность, но затемняет его причиняющую природу. С другой стороны, различение касается и фокуса исследовательского внимания: каузальная модель исходит из учета процессуально-событийной стороны, а пропозитивная – из набора участников события.

Построим конструкцию знака «Действие» (рисунок 1.8) на основе эксплицитной модели действия [38], объединяющей обе описанные выше модели:

$$Action = <a, D, P, SO, C, I, A, E, M>, \quad (1.2)$$

где a – имя действия;

SO – субъект или объект действия;

I – инструмент, с помощью которого осуществляется действие;

A – множество действий, находящихся в квалитативных отношениях с действием a;

E – событие, при наступлении которого выполняется действие, D, P, C, M имеют значение, аналогичное соответствующим элементам знака “понятие”, т.к. действия, как и понятия, могут составлять иерархии. Они могут состоять из отдельных процедур, процедуры из операций. Они могут иметь и виды или варианты.

В соответствии с конструкцией знака, показанной на рисунке 1.8, словарная статья «Действие» описывается следующим образом.

a. $a=<a_1, a_2, a_3>$, a_1 – имя термина, a_2 – тип концептуального объекта «Действие», a_3 – вид действия, выбирается из списка: процесс, процедура, функция, операция, способ, метод.

D. $D= \{D_i | D_i$ – субстанциальная дефиниция, $i=1 \div n$, n – количество выбранных дефиниций$\}$.

P. $P = \{(p_1, p_2)_i \mid p_1$ – имя свойства; p_2 – ссылка на словарную статью, описывающую p_1; i – количество свойств действия$\}$.

SO. $SO =<SO_1, SO_2, SO_3>$, SO_1 – тип: объект или субъект; SO_2 – имя термина, являющегося объектом или субъектом действия; SO_3 – ссылка на словарную статью, описывающую термин SO_2.

I. $I=<I_1, I_2>$, I_1 – имя термина, идентифицирующего инструмент, с помощью которого осуществляется действие, I_2 – ссылка на словарную статью, описывающую термин I_1.

E. $E=<E_1, E_2>$, где E_1 – имя события, при наступлении которого выполняется данное действие; E_2 – ссылка на словарную статью, описывающую термин E_1.

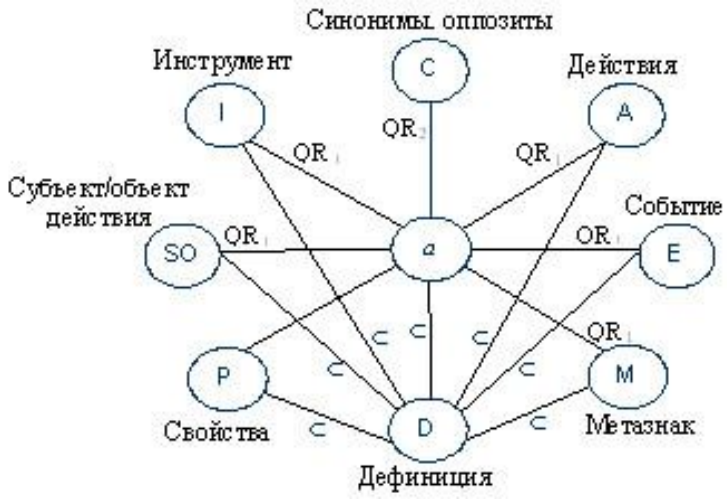


Рисунок 1.8 – Графическая интерпретация знака “Действие”: *a* – действие, QR_1 - квалитативные отношения, QR_2 – квантитативные отношения, $\subset$ - отношение включения

Все данные о действии, такие, как параметры действия, начальный объект или исходные данные действия, конечный объект или результаты действия, производный объект или промежуточные данные действия, место действия и т.п., описываются в элементе *P* – свойства действия.

1.3.3 Структуры словарных статей «*Состояние*» и «*Событие*»

Категории «Действие», «Состояние» и «Событие» отвечают за представление динамической части модели предметной области. В работах лингвистов их относят к функционально-семантическим категориям динамичности [35]. В дисциплинах точных наук «Состояние» и «Событие» понимаются как одни из существенных категорий объективной действительности. При этом категория «Состояние» представляет внешнее положение объекта, а изменчивость материального пространства определяет категория «Событие». Событиями принято называть мгновенные ситуации, состоящие в переходе от одного состояния к другому.

События противопоставляются действиям, основным свойством которых является наличие длительности, то есть способность занимать значительные отрезки временной оси. Это свойство сближает действие с состоянием, которое также обладает длительностью. Отличие действия от состояния заключается, прежде всего, в том, что для поддержания действия требуется некоторый постоянный приток энергии, без которого оно продолжаться не может, а состояние длится, как бы, «само по себе». С помощью категории действия выражается процесс изменения и развития вещей и явлений, который, в конечном итоге, сводится к изменению их

свойств и отношений. Совокупность таких свойств и отношений определяет сущность вещи или явления. Состояние охватывает все сущее в мире, выделяя вещь на основании ее всевозможных характеристик. Универсальность и всепричастность этой категории формирует онтологический статус состояния [37].

Таким образом, вторым важным отличием действия от состояния является то, что состояния никогда не описывают изменений, а действия приспособлены именно для описания различных изменений во времени. Это свойство действий иногда называют «негомогенностью», имея в виду, что они соотносятся с ситуациями, состоящими из качественно разнородных временных фаз (так, например, масса горящих дров с течением времени необратимо уменьшается). При этом среди негомогенных действий особо выделяется одна группа действий, называемых *предельными*, которые описывают такое изменение, которое не может продолжаться бесконечно и при нормальном развитии событий завершается некоторым заранее известным финалом. Так, финалом предельного процесса *гореть* будет, естественно, полное сгорание объекта, финалом предельного процесса падать – событие *упасть* и т.п. Это свойство предельных действий отчасти сближает их с событиями: и те и другие описывают переход от одного состояния к другому, но только события – мгновенный переход, а предельные процессы – длительный и многоступенчатый.

Таким образом, введение в категориальный аппарат действия, состояния и события позволяет описать в онтологии активную семантику предметной области.

Состояние свойственно объекту, поэтому первая ссылка на эту категорию принадлежит конструкции знака «Понятие». Оно связано с действием или действиями, которыми оно поддерживается, и событием, на основе которого объект перешел в это состояние. Так как данная категория обладает дефиницией и, возможно, обладает синонимами и коррелятами, метаданными и свойствами, например, длительностью, то необходимо ввести в конструкцию знака «Состояние» (рисунок 1.9) вершины *D*, *C*, *M*, *Pr*.

В соответствии с конструкцией знака, показанной на рисунке 1.9, словарную статью «Состояние» можно представить в виде следующего кортежа:

$$State = <s, D, P, E, C, M>, \qquad (1.3)$$

где

s. $s=<s_1, s_2>$, s_1 – имя термина, s_2 – тип концептуального объекта «Состояние»;

D. $D= \{D_i | D_i$ – субстанциальная дефиниция, $i=1 \div n$, n – количество выбранных дефиниций$\}$;

P. $P = \{(p_1, p_2)_i \mid p_1$ – имя свойства; p_2 – ссылка на словарную статью, описывающую p_1; i – количество свойств состояния$\}$;

E. $E=<E_1, E_2>$, E_1 – имя события, на основании которого объект переходит в данное состояние; E_2 – ссылка на словарную статью, описывающую термин E_1.

Элементы *C*, *M* аналогичны соответствующим элементам словарной статьи «Понятие».

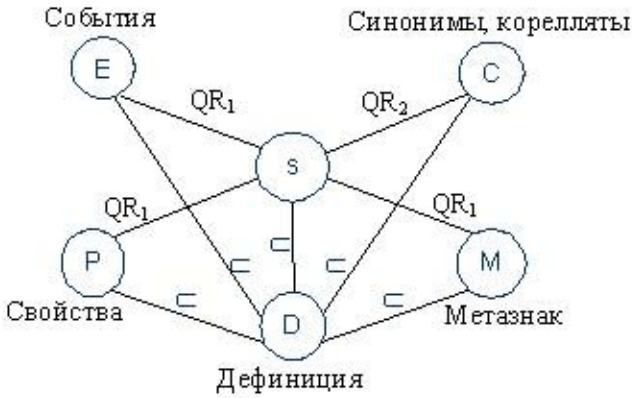


Рисунок 1.9 – Графическая интерпретация знака “Состояние”: *s* – состояние, QR_1 – квалитативные отношения, QR_2 – квантитативные отношения, ⊂ – отношение включения

Событие выступает как основание или причина действия, то есть события и действия могут находиться в причинно-следственных отношениях [111].

Знак «Событие» (рисунок 1.10) описывается как:

$$Event = <e, D, P, Y, C, S, M>, \quad (1.4)$$

где *e* – имя события;

P – множество свойств события *e*;

Y – условие, при истинности которого наступает событие *e* и которое определяет причинно-следственное отношение между событием и действием;

S – состояния объекта; *D*, *P*, *C*, *M* имеют значение, аналогичное соответствующим элементам знака “понятие”.

e. $e=<e_1, e_2>$, e_1 – имя термина, e_2 – тип концептуального объекта «Событие».

S. $S=<S_1, S_2>$, S_1 – текущее состояние объекта, S_2 – новое состояние, в которое переводится объект при наступлении события *e*;

Y. $Y=<Y_1, Y_2, Y_3>$, Y_1 – обозначение условия наступления события; Y_2 – определение условия; Y_3 – тип определения условия: лингвистическое, в виде отношения (формулы).

Рисунок 1.10 – Графическая интерпретация знака “Событие”: *e* – событие, QR_1 - квалитативные отношения, QR_2 – квантитативные отношения, ⊂ – отношение включения

В свойствах события возможно описание таких величин, как периодичность наступления (появления) события, время наступления и другие.

1.3.4 Структуры словарных статей «*Свойства*» и «*Величины*»

Категория «Свойство» выражает такую сторону понятия, которая обусловливает его различие или общность с другими понятиями и обнаруживается в его отношении к ним [97]. Всякое свойство относительно: свойство не существует вне отношений к другим свойствам и вещам. Свойство вещей (сущностей) внутренне присуще им, существует объективно, независимо от человеческого сознания. Таким образом, категория «Свойство» является одной из основных категорий, необходимых при описании знаний предметной области.

Знак «Свойство» (рисунок 1.11) задается как:

$$Property = <p, D, K, Q, C, M>, \qquad (1.5)$$

где p – имя свойства;

K – вид свойства;

Q – величины;

D, C, M имеют значение, аналогичное соответствующим элементам знака “понятие”.

Рисунок 1.11 – Графическая интерпретация знака “Свойство”: p – свойство, QR_1 – квалитативные отношения, QR_2 – квантитативные отношения, $\subset$ – отношение включения

p. $p = <p_1, p_2>$, p_1 – имя свойства; p_2 – тип концептуального объекта «Свойство»;

K. $K=<k_1, k_2>$, k_1 – вид свойства: количественное, качественное; k_2 – вид свойства: показатель, параметр, фактор, признак, контрпризнак, критерий, индикатор, данное и другие;

Q. $Q = <q_1, q_2>$, q_1 – имя величины, q_2 – ссылка на словарную статью, описывающую q_1.

Так как каждое свойство измеряется, то необходимо в категориальный аппарат ввести категорию «Величины», которая позволит описать размерность свойств.

На рисунке 1.12 показана графическая интерпретация этого знака, в соответствии с которой словарная статья «Величины» описывается в виде шестерки:

$$Quantity = <q, D, MS, L, U, M>, \quad (1.6)$$

где элементы D, M кортежа имеют такой же смысл, как и для других словарных статей;

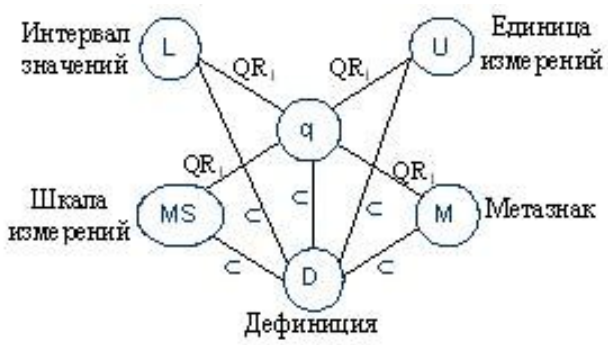


Рисунок 1.12 – Графическая интерпретация знака “Величины”: q – величина, QR_1 – квалитативные отношения, $\subset$ – отношение включения

q. $q = <q_1, q_2>$, q_1 – имя величины, q_2 – тип величины: временная (длительность), пространственная (длина, объём и т.д.) и другие величины;

MS. $MS = <MS_1, MS_2>$ – шкала измерения:

$MS_1 = <MS_{11}, MS_{12}>$, MS_{11}– качественная (лингвистическая) шкала, MS_{12} – семейство множеств терминов значений лингвистической шкалы.

$MS_2 = <MS_{21}, MS_{22}>$ – количественная шкала, MS_{21} – нижнее значение шкалы измерений, MS_{22} – верхнее значение.

L. $L = <L_1, L_2>$, L_1 – нижнее нормальное значение; L_2 – верхнее нормальное значение.

U. U – единица измерений, в случае качественной шкалы имеет значение null.

Следует отметить, что структуры словарных статей не претендуют на полную завершенность, они могут изменяться с течением времени. Поэтому способ представления модели должен носить декларативный характер, чтобы существовала возможность доступа инженеру по знаниям для её модификации.

1.4 Модель представления словарных статей

В соответствии с вышеизложенным, знак как исходный элемент семиотической системы, включает в себя три аспекта: имя знака, или синтаксический аспект знака; содержание знака, или семантический аспект знака; назначение знака, или прагматический аспект знака. Все они отражаются в словарных статьях тезауруса. Однако прагматический аспект отражается только в описательном виде, поэтому для реализации тезауруса лучше всего подходит фреймовая модель представления знаний, так как в этом случае прагматический аспект отражается в присоединенных процедурах [88].

1.4.1. Знак - фрейм

Имя фрейма соответствует имени знака (имени словарной статьи), имена обычных слотов соответствуют содержанию знака, которое описано в элементах словарной статьи, а слоты, содержащие в качестве значений имена присоединенных процедур, соответствуют прагматике знака, содержащейся в дугах графового представления знака.

Имя в знаке, как правило, несет информацию о специальном виде связей между знаками. Это – связи наследования, так они называются в теории фреймов и объектно-ориентированном проектировании. В тезаурусе они соответствуют квалитативным отношениям. Наличие квалитативных отношений в виде вертикальных и горизонтальных связей между терминами приводит к тому, что система знаков (словарных статей) оказывается объединенной в иерархическую сеть. Вершинами этой сети являются отдельные знаки, а дуги сети характеризуют различные отношения, к которым принадлежат знаки. Такие конструкции в работах по искусственному интеллекту называются семантическими сетями [36,84,85]. Каждая вершина сети представляет собой фрагмент *сети*.

Аналогия между сетью знаков и семантической сетью неслучайна. Не фиксируя явно источник своих конструкций, специалисты, работающие в области ИИ, вводят многие понятия и модели, тесно связанные с семиотикой. Одним из таких понятий является «фрейм». М. Минский впервые ввел это понятие в научный оборот в 70-х годах [46] и трактовал его как минимальное описание некоторой сущности. Причем в описании содержится все, что нужно для идентификации этой сущности. В такой интерпретации фрейм напоминает толкование понятия в треугольнике Фреге, а также концептуальных объектов, описанных в разделе 1.2. С позиций программирования понятие фрейм определяется как некоторая конструкция, описывающая сущность. В простейшем случае такая конструкция имеет следующий вид [51]:

$$N : \{<S_1, V_1, P_1>, \ldots, <S_n, V_n, P_n>\}, \qquad (1.7)$$

где

N – имя фрейма;

S_i – имя слота,

V_i – значение слота,

P_i – процедура.

Структура, число слотов, их содержание и имена могут меняться в зависимости от типа концептуального объекта и конкретной сущности, описываемой этим объектом. Каркас фрейма, в котором места, предназначенные для конкретных значений, пустуют, принято называть *фреймом-прототипом* (или *протофреймом*). Фрейм-прототип явно связан с центральной вершиной графа концептуального объекта. Информация, хранящаяся во фрейме, задает понятие, которое характеризует все сущности с данным именем. В решаемой задаче будем иметь шесть базовых фреймов-прототипов в соответствии с шестью концептуальными объектами,

конструкциями знака и словарными статьями. Они будут играть роль инвариантов. Для каждой предметной области на их основе должны быть разработаны варианты фреймов-прототипов.

Если же все слоты во фрейме имеют значения, то он несет информацию о вполне конкретной реализации сущности. Поэтому такой фрейм принято называть *фреймом-экземпляром* (или *экзофреймом*). Экземпляры связаны со знаками через представления, т.к. именно представления в конструкциях знаков связаны с конкретным проявлением сущностей. В работе сеть *экзофреймов* будет представлять собой семиотическую базу фактов по предметной области.

Таким образом, представления знаков в компьютерных системах подобны фреймовым представлениям, а сети знаков моделируются сетями фреймов, которые, в самом общем случае, представляют собой семантические сети.

Итак, в контексте прикладной семиотики структуру знака можно сопоставить со структурой фрейма. Имя знака соответствует имени фрейма. Концепт отражается протофреймом. Представление (денотат) соответствует экзофрейму. Треугольник «Имя – Протофрейм – Экзофрейм» называется знаком-фреймом [88,89]. Со знаком-фреймом связан набор базовых процедур, позволяющих вести обработку представленных знаний. К ним относятся процедуры различных видов поиска и отображения информации.

Рассмотрим понятие активности отдельного знака-фрейма или некоторого фрагмента сети, в вершинах которой находятся знаки-фреймы. Хорошо известно, что при решении задач обработки данных активную роль играют программы решения задач, а данные, с которыми работает программа, играют пассивную роль в процессе вычислений. Их активизация происходит лишь при выполнении программы в момент, когда реализуется вызов из памяти и обработка.

Для поддержки активности фрагментов сети знаков-фреймов в графовое представление знака введена вершина «Действие», для которой на следующем уровне создан фрагмент сети, имеющий вершину «Способ метаязыкового представления». Последняя вершина имеет метауровень по отношению к уровням описанной иерархической сети знаков-фреймов. Этот метауровень должен содержать метазнак, в котором описан метод обработки данной сущности в виде продукций [36]. Ввод текущей ситуации позволит активизировать данный метод, а точнее, метазнак. Таким образом, метазнаки в отличие от знаков несут в себе «заряд активности».

1.4.2. Базовые фреймы-прототипы

Фрейм может быть декларативного, процедурного и процедурно-декларативного типа. В нашем случае будем использовать фреймы процедурно-декларативного типа. В таблице 1.1 приведено соответствие слотов протофрейма и элементов словарной статьи. Прежде чем рассмотреть структуру протофрейма, необходимо определиться с набором присоединенных процедур.

Таблица 1.1 – Соответствие слотов фрейма-прототипа и элементов словарной статьи для знака «Понятие»

Имена фрейма и слотов	Значение слота	Имя словарной статьи
{<Имя термина>	Значение	t_1
<Тип концептуального объекта>	«Понятие»	t_2
<Вид сущности>	«Матер.» | «Нематер.»	t_3
(<Дефиниции>		$D = \{d_i \mid i=1..n\}$
(<Дефиниция>))	Значение	d_i
(<Свойства>		$P = \{(p_1, p_2)_i\}$
(<Имя свойства>	Значение	p_1
<Ссылка>))	Указатель	p_2
(<Действия>		$A = \{(a_1, a_2)_i\}$
(<Имя действия>	Значение	a_1
<Ссылка>))	Указатель	a_2
(<Квантитативные отношения>		$C = <C_1, C_2>$
(<Синонимы>		$C_1 = \{(c_{11}, c_{12})_i\}$
(<Синоним>	Значение	c_{11}
<Ссылка>))	Указатель	c_{12}
(<Корреляты>		$C_2 = \{(c_{21}, c_{22})_i\}$
(<Коррелят>	Значение	c_{21}
<Ссылка>)))	Указатель	c_{22}
(<Состояния>		$S = \{(s_1, s_2)_i\}$
(<Состояние>	Значение	s_1
<Ссылка>))	Указатель	s_2
(<Квалитативные отношения>		$T = <T_1, T_2, T_3, T_4>$
(<РодВид>		$T_1 = <T_{11}, T_{12}>$
(<Род>	Значение	$T_{11} = \{(t_{111}, t_{112})_i\}$
<Ссылка>)	Указатель	
(<Вид>	Значение	$T_{12} = \{(t_{121}, t_{122})_i\}$
<Ссылка>))	Указатель	
(<ЦелоеЧасть>		$T_2 = <T_{21}, T_{22}>$
(<Целое>	Значение	$T_{21} = \{(t_{211}, t_{212})_i\}$
<Ссылка>)	Указатель	
(<Часть>	Значение	$T_{22} = \{(t_{221}, t_{222})_i\}$
<Ссылка>))	Указатель	
(<Способы представления>		$T_3 = \{(t_{31}, t_{32})_i\}$
(<Способ представления>	Значение	t_{31}
<Ссылка>))	Указатель	t_{32}
(<Способы выражения>		$T_4 = \{(t_{41}, t_{42})_i\}$
(<Способ выражения>	Значение	t_{41}
<Ссылка>)))	Указатель	t_{42}
(<Отношения между знаковыми системами>		$M = <M_1, M_2>$
(<Способ метаязыкового представления>		$M_1 = \{(m_{11}, m_{12})_i\}$
(<Способ метаязыкового представления>	Значение	m_{11}
<Ссылка>))	Указатель	m_1
(<Термин другого языка>		$M_2 = \{(m_{21}, m_{22})_i\}$
(<Термин>	Значение	m_{21}
<Ссылка>))))}	Указатель	m_{22}

Во фреймах процедурно-декларативного типа процедуры привязываются к слоту путем указания последовательности выполняемых операций. Присоединенные процедуры делятся на внутренние и внешние. Внутренняя процедура используется для работы с содержимым слота фрейма, в то время как внешняя – для работы со множеством фреймов.

Внутренние процедуры активизируются при обращении к соответствующему слоту или фрейму и нацелены на обработку отдельных слотов. В рассматриваемой фреймовой модели определены следующие внутренние процедуры:

– процедуры, формирующие содержимое соответствующих слотов фрейма, например, *FunDef* – процедура, формирующая множество дефиниций термина; *FunProp* – процедура, формирующая множество свойств термина и т.д.;

– процедура *FunAct*, реализующая действие, описанное в слоте.

Внешние процедуры активизируются при возникновении определенного события. Они нацелены на обработку отдельного фрейма или сети фреймов в целом. К таким процедурам относятся:

– процедура *SearchTerm*, осуществляющая поиск по образцу в тезаурусе отдельных терминов, элементов словарной статьи;

– процедура *GenDictEntry*, формирующая на основе содержимого слотов фрейма словарную статью тезауруса в текстовом виде;

– процедура *GenThes*, осуществляющая составление тезауруса в целом в текстовом виде;

– процедура *Visual*, осуществляющая визуализацию сети знаков-фреймов или ее фрагмента;

– процедура *ActivProc*, активизирующая указанные процедуры для выполнения.

Рассмотрим структуру протофрейма на примере структуры фрейма-прототипа для концептуального объекта «Понятие»:

```
<!-- Файл Frame_#.xml -->
<?xml version='1.0'?>
<FRAME NAME="Frame_#.xml">
  <NAMETERM VALUE="…"/>
  <SLOT NAME="NameCO" VALUE="Понятие"/>
  <SLOT NAME="KindEntity" VALUE="Материальный, Нематериальный"/>
   <SLOT NAME="definitions">
    <SLOT NAME="definition" VALUE="…"/>
    <SLOT NAME="attachproc" VALUE="FunDef"/>
  </SLOT>
  <SLOT NAME="propeties">
    <SLOT NAME="property">
      <SLOT NAME="nameprop" VALUE="…"/>
      <SLOT NAME="link"      FILE="Frame_#.xml"/>
    </SLOT>
    <SLOT NAME="attachproc" VALUE="FunProp"/>
  </SLOT>
  <SLOT NAME="actions">
    <SLOT NAME="action">
      <SLOT NAME="nameact" VALUE="…"/>
      <SLOT NAME="link"     FILE="Frame_#.xml"/>
```

```
    </SLOT>
    <SLOT NAME="listattachproc">
      <SLOT NAME="attachproc" VALUE="FunAct"/>
      <SLOT NAME="attachproc" VALUE="ActivProc"/>
    </SLOT>
  </SLOT>
  <SLOT NAME="kvantrel">
    <SLOT NAME="synonyms">
      <SLOT NAME="synonym">
        <SLOT NAME="namesyn" VALUE="…"/>
        <SLOT NAME="link"    FILE="Frame_#.xml"/>
      </SLOT>
      <SLOT NAME="attachproc" VALUE="FunSyn"/>
     </SLOT>
    <SLOT NAME="correlates">
      <SLOT NAME="correlate">
        <SLOT NAME="namecorr"  VALUE="…"/>
        <SLOT NAME="link"      FILE="Frame_#.xml"/>
      </SLOT>
      <SLOT NAME="attachproc" VALUE="FunCorr"/>
    </SLOT>
  </SLOT>
  <SLOT NAME="states">
    <SLOT NAME="state">
      <SLOT NAME="namestate"    VALUE="…"/>
      <SLOT NAME="link"         FILE="Frame_#.xml"/>
    </SLOT>
    <SLOT NAME="attachproc" VALUE="FunState"/>
  </SLOT>
  <SLOT NAME="kvalitrel">
    <SLOT NAME="ClassKind">
      <SLOT NAME="Class">
        <SLOT NAME="nameclass" VALUE="…"/>
        <SLOT NAME="link"      FILE="Frame_#.xml"/>
        <SLOT NAME="attachproc" VALUE="FunClass"/>
      </SLOT>
      <SLOT NAME="Kind">
        <SLOT NAME="namekind"  VALUE="…"/>
        <SLOT NAME="link"      FILE="Frame_#.xml"/>
        <SLOT NAME="attachproc" VALUE="FunKind"/>
      </SLOT>
    </SLOT>
    <SLOT NAME="WholePart">
      <SLOT NAME="Whole">
        <SLOT NAME="namewhole" VALUE="…"/>
        <SLOT NAME="link"      FILE="Frame_#.xml"/>
        <SLOT NAME="attachproc" VALUE="FunWhole"/>
      </SLOT>
      <SLOT NAME="Part">
        <SLOT NAME="namepart"  VALUE="…"/>
        <SLOT NAME="link"      FILE="Frame_#.xml"/>
        <SLOT NAME="attachproc" VALUE="FunPart"/>
      </SLOT>
    </SLOT>
    <SLOT NAME="PresMethods">
      <SLOT NAME="PresMeth">
        <SLOT NAME="namemeth"  VALUE="…"/>
        <SLOT NAME="link"      FILE="Frame_#.xml"/>
```

```
        </SLOT>
        <SLOT NAME="attachproc" VALUE="FunPresMeth"/>
      </SLOT>
      <SLOT NAME="Styles">
        <SLOT NAME="Style">
          <SLOT NAME="namemeth"  VALUE="…"/>
          <SLOT NAME="link"      FILE="Frame_#.xml"/>
        </SLOT>
        <SLOT NAME="attachproc" VALUE="FunStyle"/>
     </SLOT>
    </SLOT>
    <SLOT NAME="metasign">
      <SLOT NAME="MetalangPresMethods">
        <SLOT NAME="MetalangPresMeth">
          <SLOT NAME="namemeth"  VALUE="…"/>
          <SLOT NAME="link"      FILE="Frame_#.xml"/>
        </SLOT>
        <SLOT NAME="attachproc" VALUE="FunMetalangPres"/>
      </SLOT>
      <SLOT NAME="TermOthLang">
        <SLOT NAME="TermOthLang">
          <SLOT NAME="namemeth"  VALUE="…"/>
          <SLOT NAME="link"      FILE="Frame_#.xml"/>
        </SLOT>
        <SLOT NAME="attachproc" VALUE="FunTOthLang"/>
      </SLOT>
    </SLOT>
    <SLOT NAME="listattachproc">
      <SLOT NAME="attachproc" VALUE="SearchTerm"/>
      <SLOT NAME="attachproc" VALUE="GenThes"/>
      <SLOT NAME="attachproc" VALUE="Visual"/>
      <SLOT NAME="attachproc" VALUE="GenDictEntry"/>
      <SLOT NAME="attachproc" VALUE="ActivProc"/>
    </SLOT>
  </FRAME>.
```

Аналогичным образом описываются протофреймы для остальных знаков. Полный вариант протофреймов приведен в Приложении А.

Как и во всяком фрейме, в знаке-фрейме легко актуализировать присоединенные процедуры, что позволяет совместить в рамках единого представления декларативную и процедурную компоненты представления знаний. Возможность организации распространения волны возбуждения по сети из-за наличия в сети возбуждающего фрагмента F делает такую модель способной к передаче активности по выбранной системе отношений.

На сети знаков-фреймов определяются операции:

– поиск фрагмента по образцу;
– замена фрагмента;
– обобщение по именам, понятиям и представлениям;
– обобщение по фрагментам сети на основе сходства-различия [90].

Эти операции основываются на логико-трансформационных правилах [89], которые в самом общем представлении имеют вид:

$$C_1;\ F_1,\ F_2,\ldots,F_k;\ \{F_i\} \Rightarrow F^*;\ C_2. \qquad (1.8)$$

Здесь C_1 играет роль условия активизации логико-трансформационного правила, $F_1,\ F_2,\ldots,F_k$ перечисляют k фрагментов сети, которые должны быть

обнаружены с помощью операции поиска фрагмента по образцу. При нахождении этих фрагментов их множество $\{F_i\}$ должно быть заменено на фрагмент F^*; после завершения этой операции выполняются постусловия C_2, которые могут вносить изменения в систему логико-трансформационных правил.

Функционирование логико-трансформационных правил напоминает действие систем продукционных правил. Разница состоит лишь в том, что в качестве объектов действий в логико-трансформационных правилах выступают фрагменты сети из знаков-фреймов.

Кроме операций с фрагментами на сетях из знаков-фреймов могут выполняться и операции вывода на знаниях и операции формирования новых фрагментов сети за счет разнообразных операций обобщения.

1.5 Выводы по главе

Анализ логико-философских и лингвистических работ, а также работ по семиотике и информатике, касающихся построения терминологии области науки, показал, что качественная терминосистема является отражением структуры области науки. Это базируется на основном свойстве термина – системности, которая является отраженным признаком системности знания, элементом которого и является термин, то есть термин представляет собой фрагмент общего смыслового единства системы. При этом значение термина как «места» в теории, как имени элемента знания проявляется в совокупности всех его системных связей с другими терминами, выясняемых на основании употребления термина в научных текстах.

Структура терминосистемы должна определять связи терминов, переходы внутри общей совокупности терминов, описывать семантику, синтактику и прагматику отдельных терминов, а также включать описание набора семантических предикатов, регулярно связывающих термины в научных текстах.

Для построения терминосистемы на основе извлечения знаний из научных текстов необходимо создать схему, которая позволит учесть свойство термина как логоса и как лексиса, то есть интегрировать в себе логико-семантические и языковые свойства термина. Наиболее адекватной такой схемой является квадрат Поспелова, содержащий метазнак и отображающий три основных аспекта знака, а именно имя знака, или синтаксический аспект знака; содержание знака или семантический аспект знака; назначение знака, или прагматический аспект знака, имеет основополагающий характер. Квадрат Поспелова как семиотическая конструкция знака имеет обобщенный характер, в то время как существуют глобальные классификации категорий объектов и отношений, которые свойственны большинству терминов. Глобальные классификации И.Дальберг представляют собой некоторые априорные схемы научного знания, которые могут накладываться на конкретную терминологию. Классификации показывают, как организуется и воплощается знание в семантической

структуре терминологии. Таким образом, конструкции знака должны их учитывать в более явном виде.

В связи с этим в данной главе на основе квадрата Поспелова и пятиугольника Никитиной разработано шесть конструкций знака в соответствии с глобальной классификацией категорий объектов, в которых учтены глобальные классификации категорий отношений.

В главе выполнено обоснование выбора тезауруса в качестве формы представления терминосистемы. Построенные конструкции знака позволили разработать структуры словарных статей тезауруса и прототипов знаков-фреймов.

Таким образом, в данной главе рассмотрены основные структуры, необходимые для построения онтологий. В следующей главе рассмотрим основные методы их построения и средства представления этих методов.

2 Методы построения онтологий предметной области

Автоматическое извлечение знаний из монологических текстов с целью построения онтологии предполагает не только выявление терминов, но и извлечение знаний о терминах. Это означает, что для описания семантической структуры терминологии необходимо распознать в тексте как термины, так и семантические отношения между терминами. Но прежде чем рассматривать типы семантических отношений, необходимо выбрать подход, на основе которого следует выполнять анализ предложения монологического текста.

2.1 Ситуационный подход в решении задач естественно-языковой обработки монологического текста

Предложение на естественном языке монологического текста представляет собой некоторое утверждение. Для распознавания семантики утверждений зачастую применяется ситуационный подход [23,83,90]. Его использование обосновывается тем, что на практике трудно создать непротиворечивую и полную базу знаний. В ситуационной семантике выводы делаются только в пределах ситуации, имеющей место в данный момент. Естественно, при переходе к другой ситуации осуществляется ревизия базы знаний, и выведенные ранее утверждения не используются для вывода новых.

При реализации ситуационного подхода за основу взята идея Л. Витгенштейна, касающаяся правил употребления слов: в зависимости от ситуаций слово, так или иначе, употребляется. Фактически им сформулировано прагматическое понимание смысла (смысл интерпретируется как адекватная реакция на возникающие ситуации). Если при этом в процессе рассуждений использовать логический вывод, то, задавая синтаксические ограничения в виде конъюнкции фактов, дополнительно описывающих ситуацию и характеризующихся двумя семантическими свойствами – непротиворечивостью и минимальностью, можно построить полную и непротиворечивую базу знаний.

В.Н. Вагин [16,17] поясняет идею Л.Витгенштейна следующим образом. Если имеется множество (контекстов) ситуаций, то имеем следующее отображение $F: S \rightarrow CONT(T)$, где S – множество ситуаций, $CONT(T)$ – множество высказываний, образующее содержание идеи, обозначаемой термом T. При этом $y=F(s) \in CONT(T)$ – аспект содержания. Таким образом, каждой ситуации s сопоставляется некоторый элемент содержания y, что соответствует нашей интуиции: когда мы употребляем некоторый термин, то реализуем инкорпорацию его содержания (иногда все из него берем, а иногда некоторую часть). Поэтому мы, как бы, реагируем на ту ситуацию, с которой имеем дело. Введем отношение предпорядка $\leq$ (транзитивное и рефлексивное отношение) на множестве $CONT(T)$. Тогда для

идеи, представимой посредством T, построена некоторая организация знания $CONT(T)$, которая создает возможность его обзора или охвата (понимания) в смысле К.И.Льюиса.

Развитие данного подхода предполагает исследование возможных ситуаций s, в которых могут находиться компоненты предложения. Возможные ситуации – это тот или иной предпорядок компонентов. Выполняя морфологический или синтаксический анализ, извлекая знания о терминах из терминологических словарей или применяя другие методы естественно-языкового анализа текста, мы всегда исследуем ситуации, в которых находятся морфемы в лексеме, лексемы – в предложении, предложение – в тексте и т.д. Таким образом, методы естественно-языковой обработки текста почти всегда направлены на анализ ситуационного контекста, и, в зависимости от представляемого метода, объектом этого анализа является либо текст, либо предложение текста, либо лексема предложения, либо морфема лексемы.

В соответствии с вышеизложенным, для решения различных задач естественно-языковой обработки монологического текста необходимо разработать методы их решения, основываясь на ситуационном моделировании. В основе ситуационного моделирования лежит простая ядерная конструкция языка $skk=xRy$, где x, y – термины, R – семантическое отношение между ними.

Рассмотрим теперь структуру продукционного правила, которое принято описывать семеркой [83,85]:

$$pr = <I, K, O, C, A \Rightarrow B, H, E>, \qquad (2.1)$$

где I – уникальное имя продукции;

K – сфера применения или секция продукции;

O – приоритетность выполнения продукции;

C – условие применимости продукции, которое обычно представляет собой логическое выражение;

$A \Rightarrow B$ – ядро продукции;

H – постдействия или постусловия продукции, имеющие вид процедур, которые выполняются в том случае, если ядро продукции реализовалось;

E – связи с другими продукциями.

Сфера применения продукций определяется характером методов. Например, при извлечении знаний о терминосистеме предметной области сферой применения K является извлечение знаний из терминологических словарей. Приоритетность продукции O проставляется автоматически по длине условия применимости, при этом самое длинное условие имеет наивысший приоритет. Постдействие H и связи E с другими продукциями определяются при разработке ядра. Основным элементом продукции является ядро продукционного правила в виде «Если A, то B». Под антецедентом A и консеквентом B понимается некоторое множество фактов.

Структура skk соответствует структуре факта. Чтобы удостовериться в этом, рассмотрим ядро продукционного правила для выявления квалитативного отношения агрегации «Часть-целое» в предложении «Аванс

составляет часть общей стоимости договора». На некотором заданном подмножестве естественного языка оно имеет следующее представление:

```
ЕСЛИ
<предложение>       p   содержит    <термин>          t1          И
<предложение>       p   содержит    <термин>          t2          И
<предложение>       p   содержит    <СемОтношение>    R           И
<СемОтношение>      r   содержит    <глагол>          v           И
<СемОтношение>      r   содержит    <терм-спутникR>   tr          И
<глагол>            v   имеет       <значение>        ["составляет"] И
<терм-спутникR>     tr  имеет       <значение>        ["часть"]   И
<термин>            t   имеет       <индекс>          i           И
<СемОтношение>      r   имеет       <индекс>          (i+1)       И
<признак>           h   имеет       <индекс>          (i+2)       И
<термин>            t2  имеет       <индекс>          (i+3)
ТО
<термин>            t1  имеет       <тип>       ["Часть"]         И
<термин>            t2  имеет       <тип>       ["Целое"]         И
<СемОтношение>      r   относится к <категория> ["ЦелоеЧасть"].
```

Данное утверждение позволяет выявить то, что в исходном предложении к категории «целое» относится термин «общая стоимость договора», а к категории «часть» – термин «Аванс», а также то, что между ними существует отношение «Целое→часть». Как видно из примера, структура простой ядерной конструкции каждого факта утверждения просматривается в явном виде, например, в первом факте «<предложение> *s* содержит <термин> t_1»: *x* = «<предложение> *s*», *R* = «содержит», *y* = «<термин> t_1».

Таким образом, в качестве модели представления метода используем продукционную модель представления знаний [26,42,119]. Это означает, что для каждого метода необходимо разработать систему продукций, которая чаще всего имеет иерархическую структуру и является декларативной формой представления метода. Например, система продукций по распознаванию семантических отношений *PrSRR* состоит из подсистем: распознавания семантических отношений «Целое→часть» *PrSRR_WP*, «Род-вид» *PrSRR_CK* и т.д.

Однако создание системы продукций – это достаточно трудоемкая задача, тем более что экспертам зачастую трудно сформулировать правила, которые они используют при решении задач, так как экспертное знание в большинстве случаев является подсознательным. Именно подсознательный характер экспертного знания вызывает трудности при построении экспертных систем, а извлечение экспертных знаний считается «узким местом» искусственного интеллекта [5,42,51]. Поэтому в работе предлагается автоматически генерировать систему продукций на основе конструктивных знаний эксперта и применения генетического программирования.

Так как продукции генерируются на ограниченном подмножестве естественного языка, то после генерации их необходимо преобразовывать в формальный вид. Проблемы генерации систем продукций и их преобразования в формальный вид рассмотрены в третьей главе.

2.2 Типизация семантических отношений

Система терминов подразумевает сложную систему семантических отношений между ними. Поэтому описание терминосистем всегда связано с классификацией не только самих терминов, но и отношений. В лингвистике классической является классификация Е.Вюстера, в которой отношения делятся на *онтологические* и *логические*. Внутри каждой категории устанавливается собственная иерархия. Логические отношения определяются как отношения подобия, онтологические – как отношения смежности в пространстве и времени. В число онтологических отношений входят отношения материала/продукта, каузальные, инструментальные, генетические. Одна из первых классификаций связей между понятиями в предметных областях предложена Л.Канделаки. Позже появилось ещё несколько глобальных классификаций терминов и отношений. Характерно, что наиболее подробные и четкие классификации ориентированы на системы информационного поиска и явились итогом соединения философско-логических исследований о природе знания с прагматическими запросами информатики. Так, возникновение одной из международных организаций по терминологии – COCTA (Conceptual and Terminological Analyis) – стимулировалось потребностями специалистов гуманитариев в логико-философском анализе своей терминологии, возникновение другой – InterConcept – потребностями в области информатики. Однако в скором времени появилась необходимость в наведении между ними мостов. Такими мостами, по мнению И. Дальберг, являются, прежде всего, единая интерпретация [27] таких метапонятий, как термин, дефиниция, референт, интенсионал и экстенсионал, концепт, категориальная классификация концептуальных объектов и их характеристик.

2.2.1 Концептуальные отношения

Следуя классификации И.Дальберг, исследуем концептуальные отношения – квалитативные и квантитативные (таблица 2.1). Рассмотрим их в разрезе понятийных сфер [82]:

- абстрактное – конкретное;
- принадлежности;
- формы и содержания;
- процессуальности;
- тождества и противопоставления.

Группировка отношений по понятийным сферам (уровням абстракции) дает возможность более четко описать семантику каждого отношения.

2.2.1.1 Квалитативные отношения

Отношения первой категории «Иерархия» делятся на три группы. Первая группа отношений принадлежит сфере «абстрактное – конкретное». В ней члены отношения соотносятся друг с другом как абстрактное и

конкретное. Иногда эту сферу называют сферой «быть» по глаголу, который связывает члены отношения:

1) род ↔ вид;
2) признак ↔ значение признака;
3) инвариант ↔ вариант.

Таблица 2.1 – Иерархия отношений между терминами

Категория отношений	Группа отношений	Отношение	Понятийная сфера
Квалитатив-ные	Иерархии	Род↔вид	Сфера абстрактного – конкретного
		Признак↔значение признака	
		Инвариант↔вариант	
	Агрегации	Целое↔часть	Сфера принадлежности
		Объект↔пространство реализации (локализации) объекта	
		Объект ↔ свойство/признак	
		Уровень↔единица уровня	
	Функциональ-ные	Объект действия ↔действие↔ субъект действия Причина↔следствие Условие↔действие Событие↔действие Состояние↔действие Событие↔состояние Инструмент↔действие Данные↔действие Данные↔величины	Сфера процессуальности
	Семиотические	Термин↔способ выражения	Сфера формы и содержания
		Термин↔способ представления	
		Термин↔метазнак термина	
Квантитатив-ные	Тождества	Термин↔синоним термина	Сфера тождества и противопоставления
	Корреляции	Термин↔коррелят термина	

Отношение «род ↔ вид» является внутрикатегориальным отношением, отношение «признак ↔ значение признака» – межкатегориальным, отношение «инвариант ↔ вариант» – общекатегориальным. В отличие от общелогических отношений «род ↔ вид» и «признак ↔ значение признака», безразличных к научным областям, в которых они реализуются, общекатегориальное отношение «инвариант ↔ вариант» в своей семантике тесно связано с теорией, описываемой в конкретном научном тексте. В общефилософском плане категории «инвариант-вариант» связаны с делением на сущность и явление.

Вторая группа отношений принадлежит сфере принадлежности. Члены отношений этой группы связаны глаголом «иметь». В нее входят отношения, являющиеся как внутрикатегориальными, так и межкатегориальными:

1) целое ↔ часть/компонент;

2) объект ↔ пространство реализации объекта (локализации или позиции);

3) объект ↔ свойство/признак;

4) уровень ↔ единица уровня.

Третья группа отношений – функциональные отношения – принадлежит сфере процессуальности. Основным отношением является отношение вида *aRb*, где *a* – объект действия, *b* – субъект действия, *R* – действие. Например, «Анализатор распознает цепочку», *a* – анализатор, *b* – цепочка, *R* – распознает. Как правило, эти отношения присущи в основном рассматриваемой предметной области и отражают её прагматику. К функциональным отношениям также относятся следующие отношения:

1) причина ↔ следствие;

2) условие ↔ действие;

3) событие ↔ действие;

4) объект действия ↔ состояние;

5) состояние ↔ действие;

6) инструмент ↔ действие;

7) данные ↔ действие;

8) данные ↔ величина.

В отличие от остальных статичных понятийных сфер сфера процессуальности является динамичной. Эти отношения позволяют описанию знака, как некоторому знанию, придать свойство активности.

Четвертая группа отношений принадлежит сфере «Форма и содержание». К этой группе относятся отношения, предназначенные для выражения соответствий между понятиями знаковых систем:

1) термин ↔ способ выражения – отражает свойства языка как системы знаков;

2) термин ↔ способ представления термина;

3) термин ↔ способ метаязыкового представления.

Посредством этих отношений можно создавать иерархии понятий знаковых систем. При этом каждый уровень иерархии будет соответствовать одному метаязыку.

2.2.1.2 Квантитативные отношения

Квантитативные отношения лежат в сфере тождества и противопоставления. К ним относятся отношения тождества (синонимии) и оппозиции (корреляции). Они симметричны, неиерархичны. Е. Вюстер делил все отношения на вертикальные, горизонтальные и диагональные. Квантитативные отношения являются горизонтальными.

Отношение синонимии – это языковое отношение различия между тождественными элементами реальности. Отношение корреляции выражает различия между понятиями и, соответственно, между объектами научного знания.

Корреляты – это термины, относящиеся к одной категории, но противопоставленные по некоторому существенному признаку.

Отношение корреляции включает:

1) отношение противоположности, или контрарное противопоставление;
2) отношение противоречия, или контрадикторное противопоставление;
3) противопоставление грамматических форм;
4) противопоставление соподчиненных видовых понятий, имеющих общее родовое понятие;
5) противопоставление частей, образующих единое целое;
6) противопоставление членов отношения соподчинения.

Контрарное противопоставление подразумевает наличие общего континуального признака, крайние точки которого образуют этот вид антонимии (например, твердость – мягкость, отрицание – утверждение). В основе контрадикторных противопоставлений лежит дихотомическое деление: однозначность – неоднозначность, значащее – незначащее, производность – непроизводность. Противопоставление грамматических форм или корреляция терминов, обозначающих грамматические формы, не относятся к антонимам. Например, по отношению противопоставления друг к другу выступают термины: мужской род, женский род и средний род; будущее, настоящее, прошедшее времена. Примером противопоставления соподчиненных видовых понятий, имеющих общее родовое понятие, является: главное предложение – придаточное предложение. Здесь родовое понятие – предложение. В качестве примеров пятого и шестого видов отношения корреляции можно привести следующие: тема – рема; управляющее – управляемое, ведущий – ведомый.

Итак, рассмотрены базовые семантические отношения и их классификация. Как упоминалось выше, семантические отношения необходимы для извлечения знаний о терминосистеме из научного текста. Поэтому необходимо рассмотреть и типы предикатов, которым они соответствуют и которые необходимы для реализации процедуры извлечения знаний.

2.2.2 Типы предикатов и семантические отношения

Категории отношений позволяют определить типы предикатов, необходимых для извлечения знаний о терминах из научных текстов. Для определения введенных отношений использованы трехместные и четырехместные предикаты, причем первый аргумент является признаком, уточняющим контекст семантического отношения, которое отражает предикат (таблица 2.2). Рассмотрим элементы кортежа предикатов.

Отношение иерархии. Данному отношению соответствует предикат PHier(a,x,y). Причем, если предикат описывает отношение "Род $\leftrightarrow$ вид", то значение первого аргумента $a \in$ {Class, Kind}, то есть если a="Род", то это означает, что переменная x обозначает имя родового понятия, а y – имя

видового понятия и наоборот. В отношении "Признак ↔ значение признака" значение первого аргумента $a \in$ {Category, Value}, в отношении "Инвариант ↔ вариант" – $a \in$ {Invariant, Variant}. Переменные x и y обозначают понятия аналогично предыдущему отношению.

Таблица 2.2 – Соответствие семантических отношений и типов предикатов

Группа отношений	Отношение	Предикат	Множество значений первого аргумента
Иерархии	Род↔вид	*PHier*(*a*,*x*,*y*)	{Class, Kind}
	Признак↔значение признака		{Category, Value}
	Инвариант↔вариант		{Invariant, Variant}
Агрегации	Целое↔часть	*PAggr*(*a*,*x*,*y*)	{Whole, Part}
	Уровень↔единица уровня	*PAggr*(*a*,*x*,*y*)	{Level, UnitLevel}
	Объект↔пространство реализации (локализации/позиции) объекта	*PExist*(*a*,*x*,*y*,*z*)	{Being, Location, Position, Order}
	Объект ↔ свойства/признак	*PProp*(*a*,*x*,*y*)	{Property, Indication, character, Parameter, Factor, Criterion}
Функциональ-ные	Объект действия ↔действие↔ субъект действия	*PFun*(*a*,*x*,*y*,z)	{Function, Causal, Condition, Event, ActState, ObjectState, Tool, Data, Quantity}
	Причина↔следствие		
	Условие↔действие		
	Событие↔действие		
	Объект действия↔состояние		
	Состояние↔действие		
	Инструмент↔действие		
	Данные↔действие		
	Данные↔величины		
Семиотические	Термин↔способ выражения	*PForm*(*a*,*x*,*y*,z)	{Expression , Representation , MetaSign}
	Термин↔способ представления		
	Термин↔метазнак термина		
Тождества	Термин↔синоним термина	*PEquiv*(*a*,*x*,*y*)	{Synonym, Quasisynonym}
Корреляции	Термин↔коррелят термина	*PCor*(*a*,*x*,*y*)	{Correlate, Oppose)

Таким образом, введение первого аргумента позволяет уменьшить количество типов предикатов, что, во-первых, является положительным фактором при использовании метода резолюций для логического вывода и, во-вторых, уточняет позицию расположения понятия относительно семантического отношения в предложении. Последнее является важным при извлечении знаний из научного текста.

Отношение агрегации. Данному отношению соответствует три типа предикатов: *PAggr*(*a*,*x*,*y*), *PExist*(*a*,*x*,*y*,*z*), *PProp*(*a*,*x*,*y*). Предикат *PAggr*(*a*,*x*,*y*) используется для определения двух видов отношений: "Целое ↔ часть" и "Уровень ↔ единица уровня". В первом отношении значение первого

аргумента $a \in$ {Whole, Part}, во втором – $a \in$ {Level, UnitLevel}, переменные x и y обозначают понятия аналогично отношениям иерархии.

Отношение "Объект ↔ пространство реализации (локализации/ позиции) объекта" включает в себя несколько отношений: существование объекта в какой-либо среде, местонахождение объекта, пространственное положение объекта, отношение порядка объекта. Всем отношениям соответствует предикат *PExist*(a,x,y,z), уточнение отношения осуществляется посредством первого аргумента предиката $a \in$ {Being, Location, Position, Order}.

Отношению "Объект ↔ пространство реализации/ существования/ проявления…" соответствует значение первого аргумента "Being", причем в кортеже предиката переменная x обозначает понятие, имя которого идентифицирует пространство, среду или объект, в котором реализован термин, обозначаемый переменной y, переменная z обозначает тип пространства реализации. Например, имеем предложение «Сом обитает в реке». Для распознавания этого предложения должен использоваться предикат *PExist*("Being",x,y,z), где x – объект существования (сом), y – среда существования (река), z = "Среда обитания". Для предложения «В глобальной сети Интернет существует очень большое количество информационных ресурсов» z="Информационное пространство", x="информационный ресурс", y = "глобальная сеть Интернет".

Отношению "Объект ↔ пространство локализации" соответствует значение первого аргумента "Location". Например, для распознавания предложения «Государство Ангола расположено в Африке» необходим предикат *PExist*("Location",x,y,z), x – объект локализации (Государство Ангола), y – место локализации (Африка), z = "Местонахождение".

Отношению "Объект ↔ позиция объекта" соответствует предикат *PExist*("Position",x,y,z), в котором x – позиционируемый объект, y – объект, относительно которого осуществляется позиционирование первого, $z \in$ {"Слева", "Справа", "Сверху", "Снизу", "Рядом", …}. Предложение «Пассажиры находятся внутри самолета» распознается предикатом *PExist*("Position",x,y,z), где z = "Внутри".

Отношению "Объект ↔ порядок объекта" соответствует предикат *PExist*("Order",x,y,z), в котором x – термин, обозначающий объект, порядок которого задан в предложении, y – термин, обозначающий объект, относительно которого определяется порядок первого, $z \in$ {"Первый", "Второй", "Последний", "Текущий", "Следующий", …}. Для предложения «После текущей темы курса рассмотрим тему «асинхронные процессы» x = "асинхронные процессы", y = "тема курса", z = "Текущая".

К группе отношений агрегации также относится отношение "Объект ↔ свойство/признак". Ему соответствует предикат *PProp*(a,x,y), в первом аргументе которого записывается имя свойства/признака или $a \in$ {Property, Indication, Character, Parameter, Factor, Criterion}, то есть в предикате уточняется о каком типе свойства идет речь: свойстве, признаке,

характеристике, параметре, факторе или критерии. Эти же слова являются дополнительными словами-признаками, определяющими данное отношение.

Функциональные отношения описываются одним предикатом. Уточнение отношения осуществляется посредством значения первого аргумента $a \in$ {Function, Causal, Condition, Event, ActState, ObjectState, Tool, Data, Quantity}. Предикат *PFun*("Function",*x*,*y*,*z*) определяет основное функциональное отношение, аргументы обозначают: *x* – объект действия, *y* – субъект действия, $z \in$ {"Метод", "Способ", "Процесс", "Процедура", "Действие", "Операция",…}.

Каузальному отношению "Причина ↔ следствие" соответствует предикат *PFun*("Causal",*x*,*y*,*z*), где *x* – термин, обозначающий причину, *y* – термин, обозначающий следствие, $z \in$ {"Из-за того, что", "В связи с тем, что", "Потому что", "Так как – то"…}.

Отношению "Условие ↔ действие" соответствует предикат *PFun*("Condition",*x*,*y*,*z*), где *x* – термин, обозначающий условие выполнения действия, *y* – действие, $z \in$ {"Если – то", "Необходимо – достаточно", "Когда – то",…}.

Отношению "Событие ↔ действие" соответствует предикат *PFun*("Event",*x*,*y*,*z*), где *x* – термин, обозначающий событие, связанное с действием, *y* – термин, обозначающий действие, $z \in$ {"Действие активизируется при", "Процесс активизируется при", "Процесс выполняется при наступлении", "После выполнения действия возникает", …}. Значения переменной *z* поясняются следующим образом: действие может выполняться после возникновения определенного события, результатом выполнения действия может быть тоже возникновение некоторого события.

Отношению "Объект действия ↔ состояние" соответствует предикат *PFun*("ObjectState",*x*,*y*,*z*), где *x* – объект действия, *y* – состояние объекта действия, $z \in$ {"Находится в состоянии", …}.

Отношению "Состояние ↔ действие" соответствует предикат *PFun*("ActState",*x*,*y*,*z*), где *x* – термин, обозначающий состояние, в котором находится действие, *y* – действие, $z \in$ {"Активное", "Пассивное","Ожидание"}.

Отношению "Инструмент ↔ действие" соответствует предикат *PFun*("Tool",*x*,*y*,*z*), где *x* – термин, обозначающий инструмент, который используется для выполнения действия, *y* – действие, $z \in$ {"С помощью", "Посредством", "С использованием", "С применением", …}.

Отношению "Данные ↔ действие" соответствует предикат *PFun*("Data",*x*,*y*,*z*), где *x* – термин, обозначающий данные, *y* – действие, $z \in$ {"Входные", "Промежуточные", "Выходные", "Статистические", "Динамические", "Временные", "Пространственные", …}.

Отношению "Данные ↔ величины" соответствует предикат *PFun*("Quantity",*x*,*y*,*z*), где *x* – термин, обозначающий величины, *y* – данное, $z \in \{T, S_1, S_2\}$, то есть значение *z* принадлежит семейству множеств, описывающих время $T=\{NT, QT\}$, пространство $S_1=\{NS_1, QS_1\}$ и положение (ситуация) $S_2=\{QS_2\}$. Так как величины могут быть как количественного

характера, так и качественного, то здесь символ *N* идентифицирует количественные величины, а символ *Q* – качественные. Примеры: *NT*={ эпоха, неделя, сутки, час, день, минута, …}, *QT*={раньше, позже, год назад, …}, NS_1 = {объем, высота, длина, ширина, координаты, диаметр, …}, QS_1 = {больше, меньше, средний, большой, низкий, высокий,…}, QS_2= {международное, семейное, официальное, социальное, материальное, чрезвычайное, осадное, безнадежное, внутриполитическое, …}.

Семиотические отношения описываются одним предикатом *PForm*(*a*,*x*,*y*,z), в котором первый аргумент уточняет отношение $a \in$ {Expression, Representation, MetaSign}.

В отношении "Термин ↔ способ выражения" значение $z \in$ {"Слово", "Предложение", "Текст"}, в отношении "Термин ↔ способ представления" значение $z \in$ {"Граф", "Рисунок", "Схема", "График", "Формула", "Модель", …} и в отношении "Термин ↔ метазнак термина" *z*=*null*. Например, предложение «Результат синтаксического анализа представляется в виде графа зависимостей» распознается предикатом *PForm*("Representation",*x*,*y*,z), где *x* = "Результат синтаксического анализа", *y* = "Граф зависимостей", *z* = "Граф". Предложение «Логус предложения можно описать формулой логики предиката первого порядка» распознается предикатом *PForm*("Representation",*x*,*y*,z), где *x* = "Логус предложения", *y*="Формулой предиката первого порядка", *z*= "Формула".

Предложение «Булева алгебра – это алгебра, несущим множеством которой является множество всех логических функций, операциями – конъюнкция, дизъюнкция, отрицание» распознается предикатом *PForm*("Expression",*x*,*y*,z), где *x*="Булева алгебра", *y*="– это алгебра, несущим множеством которой является множество всех логических функций, операциями – конъюнкция, дизъюнкция, отрицание", *z*="Предложение".

Отношение тождества или синонимии определяется предикатом *PEquiv*(*a*,*x*,*y*), *отношение корреляции* – *PCor*(*a*,*x*,*y*). В первом случае $a \in$ {Synonym, Quasisynonym}, во втором – $a \in$ {Correlate, Oppose}.

Для распознавания тех или иных отношений между терминами в научном тексте необходима дополнительная информация об этих отношениях в виде термов-спутников или термов-признаков отношений, составляющих зачастую устойчивые словосочетания с глаголом семантического отношения. В таблице 2.3 приведены примеры значений кортежа предикатов для некоторых семантических отношений.

Знание описанной информации позволяет построить диагностирующие конструкции (конструкты) по распознаванию каждого вида семантического отношения и на их основе сгенерировать множество ядер продукционных правил для извлечения знаний о терминах из научного текста.

Таблица 2.3 – Примеры значений кортежа предикатов

Отношение	Терм-спутник	*x*	*R*	Терм-спутник	*y*	Первый аргумент, *a*
Род(A)↔вид(B)		A	имеет	виды	$B_1,\ldots,B_n$	Class
	к видам	A	относится		$B_1,\ldots,B_n$	Class
	к	A	относится		$B_1,\ldots,B_n$	Class
		A	относится		$B_1,\ldots,B_n$	Class
		B	относится	к семейству	A	Kind
		B	относится	к классу	A	Kind
		B	относится	к роду	A	Kind
	родом	B	является		A	Kind
		B	принадлежит		A	Kind
Признак (A)↔ значение признака (B)		A	есть		B	Category
		A	–		B	Category
		A	имеет	имя	B	Category
		A	именуется		B	Category
		A	называется		B	Category
		A	имеет	значение	B	Value
		B	имеет	категорию	A	Category
Инвариант (A) ↔вариант (B)	к	A	относится		$B_1,\ldots,B_n$	Invariant
	инвариантом	A	является		B	Invariant
Целое (A) ↔ часть (B)	целое	A	–	часть	B	Whole
	целое	A	имеет	часть	B	Whole
		A	состоит из		B-ов	Whole
		A	включает в	как часть	B	Whole
		A	включает в		B	Whole
		A	имеет	своими частями	$B_1,\ldots,B_n$	Whole
		B	входит	в состав	A	Part
		B	составляет	часть	A	Part
		B	является	частью	A	Part
		B	является	элементом	A	Part
Объект (A) ↔ свойство/признак (B)		A	обладает	свойством	B	Propety
		A	имеет	существенным признаком	B	Indication
	для	A	характерно		B	Character
		A	характеризует	наличием	B	Character
	фактора	A	являются		$B_1,\ldots,B_n$	Factor

Ядром продукционного правила является предикат. Чтобы создать набор конструктов для каждого метода, распознающего тот или иной объект исследования, необходимо знать суть самих методов решения данных задач.

2.3 Обобщенная схема естественно-языковой обработки монологического текста

Методы построения онтологий условно разделим на группы. В первую группу войдут традиционные методы естественно-языковой обработки текста, во вторую – методы, касающиеся непосредственно построения онтологии.

Рассмотрим технологию анализа естественно-языкового текста, предложенную в работах [10,36,55,61]. Модифицированный вид обобщенной схемы приведен на рисунке 2.1.

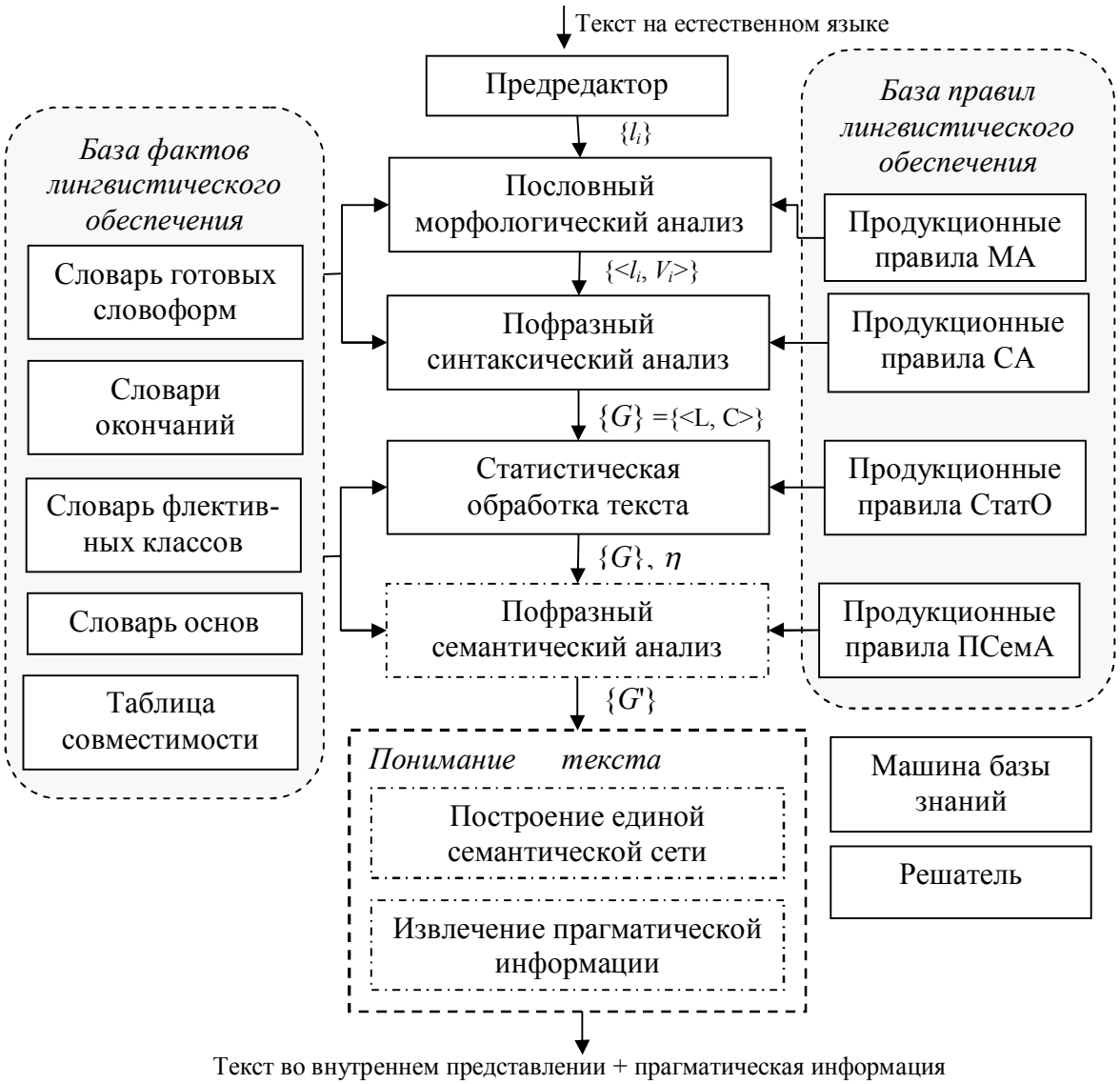


Рисунок 2.1 – Обобщенная схема анализа монологического текста

Методы реализации первых четырех блоков считаются наиболее проработанными. Однако надо заметить, что исследования продолжаются, так как удовлетворительный результат их работы пока не получен.

В функции *предредактора* входят лексический анализ, разбиение сложных предложений на простые, выделение в исходном тексте разделов, подразделов, предложений, проверка выполнения принятых ограничений. На настоящем этапе исследований научных текстов недопустимыми считаются сложноподчиненные предложения, включающие рекурсивно-вложенные определительные предложения.

Основная задача лексического анализа заключается в разбиении входного текста документа, представляющего собой последовательность одиночных символов, на последовательность лексем [36, 66,79]. Все символы

входной последовательности с этой точки зрения разделяются на символы, принадлежащие каким-либо лексемам, и символы, разделяющие лексемы (разделители). В некоторых случаях между лексемами может и не быть разделителей. В результате лексического анализа формируется множество лексем $L = \{l_i \mid i=1 \div k,\ k$ – количество лексем в тексте$\}$. Каждой лексеме приписывается вектор:

$$\rho_i = < p_i,\ n_i^l,\ n_i^s,\ n_i^p,\ n_i^d,\ n_i^c >, \qquad (2.2)$$

где p_i – уникальный номер вектора лексемы;

n_i^l – порядковый номер лексемы в предложении;

n_i^s – порядковый номер предложения в тексте;

n_i^p – номер параграфа; n_i^d – номер раздела;

n_i^c – номер главы.

Основной функцией *пословного морфологического анализа* является определение части речи лексемы l_i и присвоение ей вектора морфологической информации ρ_i. При анализе лексем используются словари окончаний, словарь флективных классов, словарь готовых словоформ и словарь основ, таблицы совместимости основы флективного класса и вектора морфологической информации.

Пофразный синтаксический анализ. В процессе синтаксического анализа должны быть однозначно определены все синтаксические единицы естественно-языкового предложения. Синтаксическими единицами называются конструкции, в которых их элементы (компоненты) объединены синтаксическими связями и отношениями. Синтаксическая связь является выражением взаимосвязи элементов в синтаксической единице, то есть служит для отображения синтаксических отношений между словами, создает синтаксическую структуру предложения и словосочетания, а также условия для реализации лексического значения слова. Обычно рассматривается только один вид синтаксической связи – подчинение. Этот вид синтаксической связи передает отношения между фактами объективного мира в виде сочетания двух слов, в котором одно выступает как главное, второе – как зависимое. Отношения между лексемами представляются в виде лексико-грамматических связей между словами, которые представляют собой вопрос от главного слова к зависимому (например, *система (какая) операционная*). Исходными данными для проведения синтаксического анализа являются результаты морфологического анализа, представленные в виде множества пар $< l_i,\ \rho_i >$, где l_i – лексема, ρ_i – вектор морфологической информации лексемы l_i. В результате проведения синтаксического анализа должен быть сформирован граф зависимостей G, в вершинах которого помещаются лексемы. Вершины соединяются дугами c, указывающими направление связи от главного слова к зависимому.

Статистическая обработка текста не является обязательной для каждой системы естественно-языковой обработки текста. Обычно она присутствует в поисковых системах и системах автоматического реферирования.

Статистические методы основаны на частотных характеристиках текста: частота вхождения слова в текст, частота совместного вхождения нескольких слов, взвешенная частота вхождения и т.д. В этих методах отношения между словами не анализируются с лингвистической точки зрения. При статистическом анализе выполняется поиск входных последовательностей слов и выделение понятий, под которыми понимаются слова и словосочетания, а также определение их частотных характеристик. Особенно важно найти субстантивные именные словосочетания, выражаемые схемой: согласуемое слово + существительное [11,76,77].

Пофразный семантический анализ. Цель семантического анализа состоит в определении для каждого слова и фразы в целом некоторых смысловых характеристик [10,36]. Содержание фразы, как правило, представляется в виде фрагмента семантической сети. Основой для построения фрагмента является граф зависимостей. Результатом семантического анализа является преобразование графа зависимостей во фрагмент семантической сети.

Блок понимания текста включает два компонента: построение единой семантической сети и извлечение прагматической информации.

Извлечение прагматической информации. В данном блоке из анализируемого текста извлекается его прагматическое содержание. Например, при информационном поиске определяется степень релевантности данного текста полученному запросу. При этом построенная сеть документа позволит более точно проранжировать найденные документы по степени их релевантности запросу.

Лингвистическое обеспечение состоит из трех частей: базы фактов, базы правил и базы знаний о предметной области. База фактов включает словарь готовых словоформ, словарь окончаний, словарь флективных классов и словарь основ. База правил состоит из продукционных правил лексического, морфологического, синтаксического, статистического и семантического анализов.

Для построения онтологической модели предметной области [58,64] представленная на рисунке 2.1 схема преобразуется в схему, представленную на рисунке 2.2.

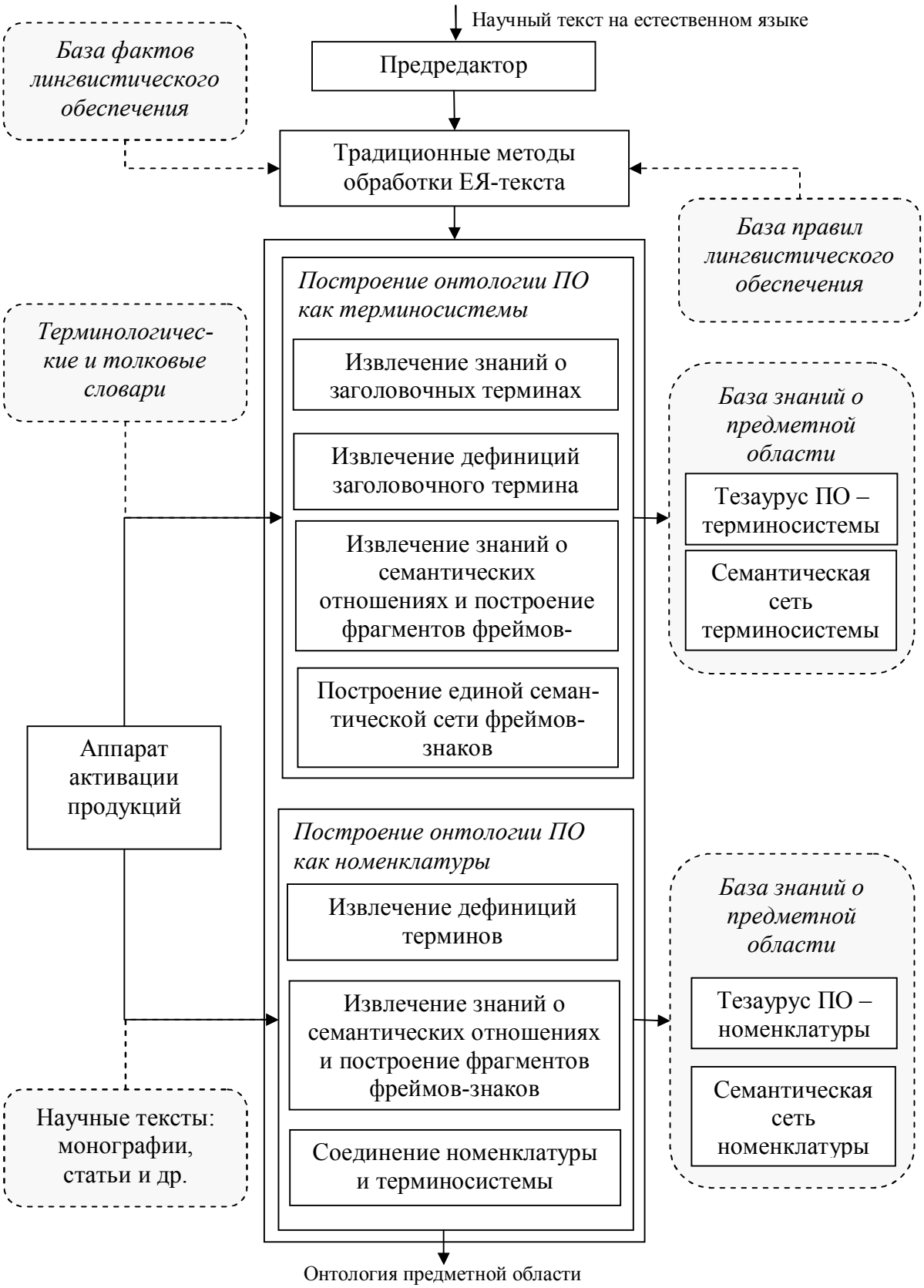


Рисунок 2.2 – Обобщенная схема построения онтологии предметной области

В этой схеме к традиционным методам анализа добавлены специальные методы построения онтологий, которые будут рассмотрены в подразделе 2.5.

2.4 Традиционные методы естественно-языковой обработки монологического текста

Из традиционных методов анализа, необходимых для построения онтологий, кратко рассмотрим морфологический, синтаксический анализы и метод выделения словосочетаний. Полное описание всех методов приведено в работах [7,32,48,49,50,52,54,70,75,98].

2.4.1 Морфологический анализ

Входной информацией морфологического анализа является множество лексем предложения, словари основ, окончаний, готовых словоформ, флективных классов, а также таблицы совместимости основы флективного класса и вектора морфологической информации.

Морфологический анализ представляется четверкой:

$$M_{MA}=<SRW, PE, ICA, GMI, RSC>, \quad (2.3)$$

где *SRW* – поиск готовой словоформы (Search Read Wordform);

PE – выделение основы и окончания (Part Extraction);

ICA – поиск флективного класса словоформы (Inflected Class Analysis);

GMI – формирование вектора морфологической информации (Generation Morphological Information);

RSC – формирование конфликтного множества (Resolution Set Collision).

Компонент SRW предназначен для поиска словоформы в словаре готовых словоформ. По готовой словоформе определяется ее флективный класс.

Компонент PE предназначен для поиска словоформы в словаре основ всех частей речи. Если поиск дал отрицательный результат, то в словоформе выделяется окончание минимальной длины (одна буква), а её остаток становится основой. Далее происходит поиск окончания в словаре окончаний различных частей речи. Если окончание не найдено, то длина окончания увеличивается на единицу, и поиск повторяется. Максимальная длина окончания равна трем буквам. При положительном результате осуществляется поиск основы в словаре основ. По основе определяется флективный класс словоформы.

Надо отметить то, что одной основе может соответствовать несколько флективных классов.

Компонент ICA. Флективный класс определяет неизменяемую морфологическую информацию о словоформе и, в первую очередь, часть речи. Для существительного флективный класс описывает также такие морфологические характеристики, как одушевленность и род словоформы.

В случае если для одной словоформы найдено несколько флективных классов, то это говорит о том, что словоформа относится к нескольким частям речи. Например, словоформа «рабочий» относится к двум частям речи: существительному и прилагательному. В этом случае формируется конфликтное множество флективных классов.

Компонент GMI. По окончанию и флективному классу по таблице совместимости определяется вектор морфологической информации. Надо отметить, что не всегда можно однозначно выявить вектор морфологической информации. Иногда возникают случаи, когда одной словоформе можно сопоставить несколько векторов морфологической информации. Например, при анализе словоформы «окно» правомерно счесть, что словоформа стоит и в именительном, и в винительном падеже. В этом случае формируется конфликтное множество векторов морфологической информации, которое разрешается на этапе синтаксического анализа.

Компонент RSC предназначен для разрешения конфликтных множеств флективных классов. Для решения этой задачи можно обойтись класическими продукционными правилами, в которых активизируется морфологическая информация словоформ.

Рассмотрим пример разрешения конфликтного множества флективных классов. Пусть дано предложение: «Это решение позволяет обеспечить целостность данных». Для словоформы «данных» этого предложения будут найдены два флективных класса, определяющих принадлежность словоформы к существительному и прилагательному.

При разрешении данного вида конфликтного множества, обозначим его K, в которое включены флективные классы F_1 (принадлежность к существительному) и F_2 (принадлежность к прилагательному), срабатывает одно из правил, которые анализируют число, род и падеж словоформы «данных» l_i и словоформы l_{i-1}, расположенной в предложении ранее рассматриваемой.

<*Продукционное правило* 1 по разрешению конфликтного множества *K*>

ЕСЛИ

<словоформа>	l_1	имеет	<индекс>	i	И
<словоформа>	l_1	имеет	<характеристика>	x_1	И
<характеристика>	x_1	есть	<число>	c_1	И
<число>	c_1	имеет	<значение>	"Ед"	И
<словоформа>	l_1	имеет	<характеристика>	x_2	И
<характеристика>	x_2	есть	<род>	c_2	И
<род>	c_2	имеет	<значение>	"Жен"	И
<словоформа>	l_1	имеет	<характеристика>	x_3	И
<характеристика>	x_3	есть	<падеж>	c_3	И
<падеж>	c_3	имеет	<значение>	"Им"	И
<словоформа>	l_2	имеет	<индекс>	$(i-1)$	И
<словоформа>	l_2	имеет	<характеристика>	x_4	И
<характеристика>	x_4	есть	<число>	c_4	И
<число>	c_1	эквивалентен	<число>	c_4	И
<словоформа>	l_2	имеет	<характеристика>	x_5	И
<характеристика>	x_5	есть	<род>	c_5	И
<род>	c_2	эквивалентен	<род>	c_5	И
<словоформа>	l_2	имеет	<характеристика>	x_6	И
<характеристика>	x_6	есть	<падеж>	c_6	И
<падеж>	c_3	эквивалентен	<падеж>	c_6	

ТО

<словоформа>	l_1	имеет	<флективный класс>	F_2

<*Продукционное правило* 2 по разрешению конфликтного множества K>

ЕСЛИ

<словоформа>	l_1	имеет	<индекс>	i	И
<словоформа>	l_1	имеет	<характеристика>	x_1	И
<характеристика>	x_1	есть	<число>	c_1	И
<число>	c_1	имеет	<значение>	"Ед"	И
<словоформа>	l_1	имеет	<характеристика>	x_2	И
<характеристика>	x_2	есть	<род>	c_2	И
<род>	c_2	имеет	<значение>	"Жен"	И
<словоформа>	l_1	имеет	<характеристика>	x_3	И
<характеристика>	x_3	есть	<падеж>	c_3	И
<падеж>	c_3	имеет	<значение>	"Им"	И
<словоформа>	l_2	имеет	<индекс>	$(i-1)$	И
<словоформа>	l_2	имеет	<характеристика>	x_4	И
<характеристика>	x_4	есть	<число>	c_4	И
<число>	c_1	не эквивалентен	<число>	c_4	И
<словоформа>	l_2	имеет	<характеристика>	x_5	И
<характеристика>	x_5	есть	<род>	c_5	И
<род>	c_2	эквивалентен	<род>	c_5	И
<словоформа>	l_2	имеет	<характеристика>	x_6	И
<характеристика>	x_6	есть	<падеж>	c_6	И
<падеж>	c_3	эквивалентен	<падеж>	c_6	

ТО

<словоформа>	l_1	имеет	<флективный класс>	F_1

<*Продукционное правило* 3 по разрешению конфликтного множества *K*>

ЕСЛИ

<словоформа>	l_1	имеет	<индекс>	i	И
<словоформа>	l_1	имеет	<характеристика>	x_1	И
<характеристика>	x_1	есть	<число>	c_1	И
<число>	c_1	имеет	<значение>	"Ед"	И
<словоформа>	l_1	имеет	<характеристика>	x_2	И
<характеристика>	x_2	есть	<род>	c_2	И
<род>	c_2	имеет	<значение>	"Жен"	И
<словоформа>	l_1	имеет	<характеристика>	x_3	И
<характеристика>	x_3	есть	<падеж>	c_3	И
<падеж>	c_3	имеет	<значение>	"Им"	И
<словоформа>	l_2	имеет	<индекс>	$(i-1)$	И
<словоформа>	l_2	имеет	<характеристика>	x_4	И
<характеристика>	x_4	есть	<число>	c_4	И
<число>	c_1	эквивалентен	<число>	c_4	И
<словоформа>	l_2	имеет	<характеристика>	x_5	И
<характеристика>	x_5	есть	<род>	c_5	И
<род>	c_2	не эквивалентен	<род>	c_5	И
<словоформа>	l_2	имеет	<характеристика>	x_6	И
<характеристика>	x_6	есть	<падеж>	c_6	И
<падеж>	c_3	эквивалентен	<падеж>	c_6	

ТО

<словоформа>	l_1	имеет	<флективный класс>	F_1

<*Продукционное правило* 4 по разрешению конфликтного множества *K*>

ЕСЛИ

<словоформа>	l_1	имеет	<индекс>	i	И
<словоформа>	l_1	имеет	<характеристика>	x_1	И
<характеристика>	x_1	есть	<число>	c_1	И
<число>	c_1	имеет	<значение>	"Ед"	И
<словоформа>	l_1	имеет	<характеристика>	x_2	И
<характеристика>	x_2	есть	<род>	c_2	И
<род>	c_2	имеет	<значение>	"Жен"	И
<словоформа>	l_1	имеет	<характеристика>	x_3	И
<характеристика>	x_3	есть	<падеж>	c_3	И
<падеж>	c_3	имеет	<значение>	"Им"	И

<словоформа> l_2 имеет <индекс> (i-1) И
<словоформа> l_2 имеет <характеристика> x_4 И
<характеристика> x_4 есть <число> c_4 И
<число> c_1 эквивалентен <число> c_4 И
<словоформа> l_2 имеет <характеристика> x_5 И
<характеристика> x_5 есть <род> c_5 И
<род> c_2 эквивалентен <род> c_5 И
<словоформа> l_2 имеет <характеристика> x_6 И
<характеристика> x_6 есть <падеж> c_6 И
<падеж> c_3 не эквивалентен <падеж> c_6
ТО
<словоформа> l_1 имеет <флективный класс> F_1

Аналогичным образом описываются правила для разрешения конфликтного множества флективных классов для частей речи: глагол или существительное, прилагательное или наречие и так далее. Тем не менее, по завершении морфологического анализа может иметь место несколько векторов морфологической информации.

2.4.2 Выделение устойчивых словосочетаний

Термин может состоять не только из одной, но и из последовательности лексем. Именно цельные именные группы, а не отдельные слова характеризуют содержание текста. Поэтому рассмотрим вопрос о возможных словосочетаниях, обозначающих термин.

Общепринятого определения понятия «устойчивое словосочетание» не существует. Некоторые авторы (например, [6]) приравнивают его к понятию «фразеологическая единица». Такое определение представляется излишне узким. Фразеологическая единица обычно трактуется как выражение, смысл которого не определяется суммой значений составляющих его слов. «Устойчивость» же фразеологизмов – фактор вторичный, напрямую не связанный с их определением. Существуют устойчивые словосочетания, не являющиеся фразеологизмами, и, наоборот, фразеологизмы, степень устойчивости которых весьма относительна. В частности, составители фразеологического словаря [44] основной упор делают, как раз, на широком изучении фразеологической вариантности и синонимии.

В настоящей работе предлагается некоторое конструктивное определение устойчивости словосочетания, апеллирующее работе [9]. В связи с тем, что задача данной работы заключается в нахождении устойчивых словосочетаний в конкретном научном тексте, то понятие устойчивости повторяющейся цепочки x, состоящей из $l>1$ подряд идущих лексем текста, не прерываемых знаками препинания, формулируется исходя из анализа лево- и правосторонних расширений всех ее вхождений в текст. Цепочка x считается устойчивой, если её частота встречаемости $F(x)>2$. В реализованном алгоритме выявления цепочек, устойчивых в указанном смысле, морфологическая вариативность учитывается путем предварительной нормализации словоформ текста. Эксперимент на научно-технических текстах подтвердил перспективность развиваемого метода. Полностью статистические методы обработки научного текста и выделение

словосочетаний описаны в работах [7,65,77]. В данном разделе рассмотрим только основные аспекты алгоритма выявления устойчивых словосочетаний в научном тексте.

Пусть l_i – произвольное понятие текста, содержащее одну лексему, тогда x_i – цепочка из k подряд следующих лексем за лексемой l_i, не прерываемых знаками препинания, $W(x_i)$ – частота совместной встречаемости лексем цепочки в тексте. Всевозможные левосторонние расширения цепочки x_i имеют форму al_i, где a – произвольная лексема, предшествующая l_i. Обозначим через a^*l_i её итеративное расширение влево. Очевидно, что $f(l_i) \geq W(a^*l_i)$. Аналогично, для всевозможных правосторонних расширений вида l_ib^* справедливо соотношение $f(l_i) \geq W(l_ib^*)$. Возможно существование комбинированных цепочек вида $a^*l_ib^*$, для которых также справедливо соотношение $f(l_i) \geq F(a^*l_ib^*)$. Цепочка x_i с $F(l_i)>1$ считается устойчивой для конкретного научного текста.

В процессе морфологического анализа подсчитывается частота вхождения $f(l_i)$ лексемы l_i, выраженной именем существительным, в текст. При этом будет создано множество $L=\{(l_i, f(l_i)) | f(l_i)>1, i=1 \div m, m$ – количество имен существительных в тексте$\}$. Далее для каждой $l_i \in L$ строятся множества левосторонних и правосторонних цепочек (словосочетаний).

Существуют различные типы и модели словосочетаний. Как правило, различают следующие лексико-грамматические типы словосочетаний: глагольные, именные, наречные. Глагольные словосочетания имеют следующие модели:

1) глагол + существительное или местоимение (с предлогом или без предлога);
2) глагол + инфинитив или деепричастие;
3) глагол + наречие.

Именные словосочетания делятся на субстантивные, адъективные, с главным словом числительным и с главным словом местоимением.

Основные модели субстантивных словосочетаний:

1) согласуемое слово + существительное;
2) существительное + существительное;
3) существительное + наречие;
4) существительное + инфинитив.

Основные модели адъективных словосочетаний:

1) прилагательное + наречие;
2) прилагательное + существительное (местоимение);
3) прилагательное + инфинитив.

Словосочетания наречного типа (с предикативными и непредикативными наречиями) имеют две модели:

1) наречие + наречие;
2) наречие + существительное.

Из приведенных классификаций видно, что для построения категорийно-понятийного аппарата научного текста необходимо

использовать модель субстантивных именных словосочетаний, выражаемую схемой: согласуемое слово + существительное. В этой модели существительное является стрежневым словом $l_i \in L$, а согласуемое слово – зависимым, и может выражаться как прилагательным, так и существительным. Будем считать терминами именные словосочетания, которые могут включать в свой состав следующие классы слов: существительные, прилагательные и сочинительные союзы. Обозначим стрежневое слово символом '*l*', существительное в родительном падеже – '*p*', прилагательное – '*r*', сочинительный союз –'*c*', тогда модели цепочек именных словосочетаний, необходимых для выделения терминов, будут иметь вид, показанный в таблице 2.4.

В тексте осуществляется поиск цепочек всех моделей, и для каждой цепочки $x_i \in X$ подсчитывается частота совместной встречаемости $F(x_i)$.

Для учета морфологической вариативности все словоформы цепочки приводятся к каноническому виду.

Таблица 2.4 – Структурный состав именных словосочетаний

№ п/п	Модель цепочки	Примеры
Левосторонние цепочки		
1	*pl*	Информационная система Образовательное сообщество
2	*ppl*	Информационная поисковая система Научное образовательное сообщество
3	*pppl*	Сетевое научное образовательное сообщество Управляющая цифровая вычислительная машина
Правосторонние цепочки		
4	*lr*	Система подготовки Сообщество детей
5	*lpr*	Система образовательных сообществ
6	*lppr*	Система научных образовательных сообществ
7	*lpppr*	Система сетевых научных образовательных сообществ
8	*lrr*	Система поиска информации Система подготовки кадров
9	*lrrr*	Система управления базами данных
10	*lrcrr*	Система хранения и поиска информации Система подготовки и переподготовки кадров
Комбинированные цепочки		
11	*plr*	Автоматический поиск информации Научная деятельность учащихся
12	*plrr*	Автоматизированная система поиска информации
13	*pplpr*	Распределенная поисковая система научной информации
14	*plrcr*	Образовательное сообщество детей и взрослых

Следует отметить, что некоторые термины текста состоят из двух словосочетаний, например, термин «образовательное сетевое сообщество по методическим вопросам» – двухсоставной. Выявить составные (композиционные) термины можно, проанализировав их совместную

встречаемость в тексте. Если они встречаются в тексте более одного раза, то можно считать, что такое сочетание терминов есть композиционный термин $y \in Y$, которое определяется как $Y = \{y_k | \ y_k = x_i \circ x_j \circ x_r;\ x_r\ i \neq j \neq r;\ i, j, r = 1 \div n,\ n$ – количество простых цепочек в тексте; $k = 1 \div q,\ q$ – количество двухсоставных терминов; x_r – возможно пустая цепочка$\}$.

Тогда общее множество терминов текста определяется по формуле:

$$Term = X \cup Y \cup (L \setminus Z), \qquad (2.4)$$

где Z – множество лексем, являющихся стержневыми словами словосочетаний.

Факт наличия композиционных терминов подтверждают лингвисты, говоря об эпистемической плотности научного текста. Согласно Т.В. Вяничевой грамматически ведущий компонент многих синлексов обладает очень общим, предельно отвлеченным лексическим значением и способен связываться с широким кругом слов, «как бы извлекающих из него самые конкретные элементы значения, что и способствует его десемантизации». Это свойство синлекса соответствует одной из ведущих черт научного стиля речи – подчеркнутой логичности, поэтому синлексы весьма широко используются в научных текстах в качестве средства логической конденсации содержания. При этом существенно отметить, что формирование нетипичных двух-, а именно трехкомпонентных синлексов представляется нам особенностью индивидуального мышления и стиля речи ученого. Изложение научного знания соотносится с развертыванием эпистемической ситуации – обычно двух- или трехаспектной, охватывающей онтологический, методологический и аксиологический аспекты знания. Реализация в тексте объединения языковых единиц, ориентированных на выражение тех или иных аспектов знания (эпистемической ситуации), именно «бесшовная» экспликация единства этих аспектов, и свидетельствует об эпистемической плотности текста. Эпистемическая плотность текста достигается посредством использования общенаучных слов широкой семантики, в том числе отглагольных существительных, в контексте ориентированных на выражение методологического аспекта знания. Такие номинации типа проблема, гипотеза, данные, экспериментальные данные, факты, наблюдения, явления; метод, способ, прием, классификация, типология, концепция, теория, закон и многие другие требуют уточнения, конкретизации в тексте, поэтому нередко притягивают к себе либо синлексикализованные термины, либо блоки связанных номинативных единиц, тем самым конденсируя, уплотняя и содержательно, и эпистемически научный текст.

2.4.3 Синтаксический пофразный анализ

Синтаксический анализ осуществляется на основе использования следующих видов языковой информации: сведений о морфологических характеристиках словоформ; сведений о соответствии между словоформами их морфологических характеристик, которые могут быть связаны

синтаксическим отношением (отношением зависимости); сведений о порядке слов в предложении; сведений о корректности результатов синтаксического анализа; сведений о пунктуации. Результат синтаксического анализа представляется в виде графа зависимостей. В соответствии с этим формальное описание модели синтаксического анализа *SA* имеет вид:

$$SA = <P, \Psi, \Pi, Q, G>, \quad (2.5)$$

где P – входная информация или предложение;

Π – промежуточное представление графа зависимостей;

Ψ – правила перехода от структуры предложения P к структурам Π;

Q – правила определения итогового графа зависимостей;

G – выходное представление результатов проведения синтаксического анализа (граф зависимостей).

В качестве входной информации выступает предложение естественного языка, которое определяется как множество кортежей:

$P=\{(l_i,\rho_i) \mid l_i \in L$, L – множество лексем предложения P, ρ_i – вектор морфологической информации, $i=1 \div n$, n – количество лексем в предложении $P\}$.

Промежуточное представление Π близко по структуре к выходному представлению G и представляет собой фрагменты графа зависимостей в виде пары вершин и отношения зависимости между ними вида «хозяин – слуга». Таким образом, каждая пара (l_i,l_j) принадлежит отношению зависимости η, то есть:

$$\Pi = \{(l_i, \eta, l_j) \mid l_i = \eta(l_j),\ i \neq j;\ i,j = 1 \div n\} \quad (2.6)$$

Отношение $l_i \eta\ l_j$ назовем простой синтаксической конструкцией. Тогда выходное представление результатов проведения синтаксического анализа G представляет собой множество простых синтаксических конструкций и имеет вид графа зависимостей:

$$G=<L, C>, \quad (2.7)$$

где L – множество вершин графа G;

C – множество дуг.

Итоговый граф зависимостей G должен быть ацикличным связным графом.

Граф зависимостей не может иметь петель, так как разработанные правила учитывают как морфологическую информацию лексем, так и порядок лексем в естественно-языковом предложении.

В случае, когда в результате морфологического анализа сформировано конфликтное множество векторов морфологической информации, синтаксическим анализатором строится несколько графов зависимостей. Поэтому возникает конфликтное множество графов зависимостей. Для разрешения конфликтного множества графов зависимостей в работе [67] предложен способ, основанный на применении нейронной сети, для определения весового коэффициента w_{ji} каждого правила синтаксического разбора. При синтаксическом анализе предложения используется набор

правил, который обладает весовым коэффициентом W_j, где j – номер графа зависимостей:

$$W_j = \sum_{i=1}^{m} w_{ji}\,, \qquad (2.8)$$

где m – количество правил, использованных при разборе предложения.

Этот коэффициент приписывается соответствующему графу зависимостей G_j. В процессе анализа выбирается тот граф зависимостей $G_j \in G_{SC}$ (G_{SC} – конфликтное множество графов зависимостей), у которого W_j является максимальным. Однако возможны случаи, когда у нескольких графов G_j вес W_j будет являться максимальным, то есть разные наборы синтаксических правил могут дать равное значение весового коэффициента W_j.

Для разрешения оставшегося конфликтного множества графов G'_{SC} зависимостей используем метод нечеткого регулирования Мамдани [4]. Для этого введем лингвистическую переменную «Степень включения синтаксических конструкций именного субстантивного словосочетания в граф зависимостей» и выходную лингвистическую переменную «Степень достоверности гипотезы о корректности графа зависимости (*DegreeValidHypothesis*)». Именное субстантивное словосочетание представляется также в виде графа зависимостей G_S как множество простых синтаксических конструкций. Компоненты нечеткого вывода рассмотрим на примере определения степени достоверности гипотезы о корректности графа зависимости G_j.

База правил нечетких продукций состоит из следующих элементов.

Имя продукции: номер нечеткой продукции.

Сфера применения: выявление корректного графа зависимости.

Условие применимости: $G'_{SC} \neq \varnothing$.

Условие ядра: составное нечеткое высказывание вида:

$$\left|G'_{SC_j} \cap G'_{S_1}\right| = t' \text{ И } \left|G'_{SC_j} \cap G'_{S_2}\right| = t' \text{ И } \ldots \text{ И } \left|G'_{SC_j} \cap G'_{S_m}\right| = t', \qquad (2.9)$$

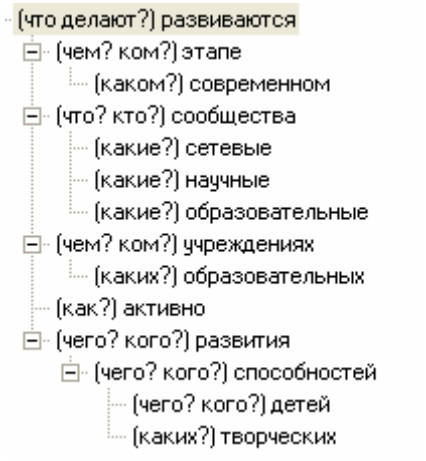
где

$\left|G'_{SC_j} \cap G'_{S_i}\right|$ – обозначение входных лингвистических переменных «СТЕПЕНЬ ВКЛЮЧЕНИЯ СИНТАКСИЧЕСКИХ КОНСТРУКЦИЙ ИМЕННОГО СУБСТАНТИВНОГО СЛОВОСОЧЕТАНИЯ В ГРАФ ЗАВИСИМОСТЕЙ»; терм $t' \in T_1$ = {Низкая (Small), Средняя (Mean), Высокая (Big)}.

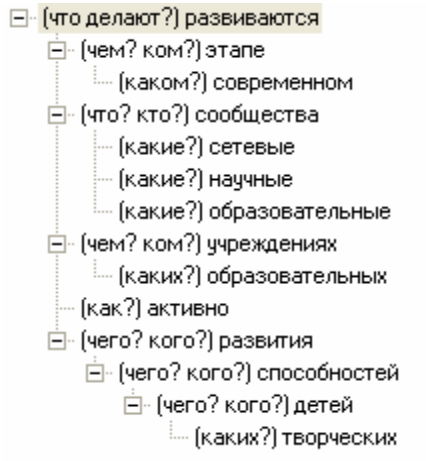
Заключение ядра: нечеткое высказывание вида «$dvh = \Delta t''$», где *dvh* – обозначение выходной лингвистической переменной: «СТЕПЕНЬ ДОСТОВЕРНОСТИ ГИПОТЕЗЫ О КОРРЕКТНОСТИ ГРАФА ЗАВИСИМОСТИ», терм $t'' \in T_2$ = {Низкая (Low), Средняя (Normal), Высокая (High)}, модификатор $\Delta \in M_2$ = {Ниже (Down), Выше (Up)}.

Рассмотрим разрешение конфликтного множества на примере разбора предложения P = «На современном этапе для развития творческих способностей детей в образовательных учреждениях активно развиваются

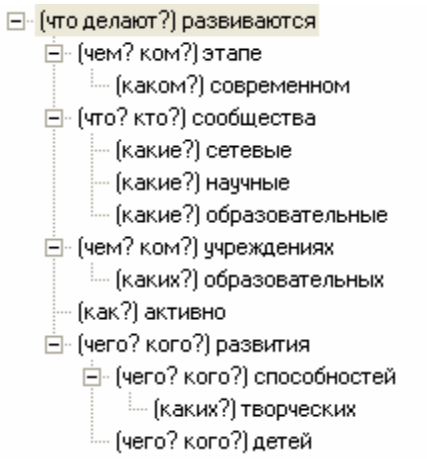
сетевые научные образовательные сообщества». В результате синтаксического анализа формируется конфликтное множество графов зависимостей G'_{SC}. На рисунке 2.3 показаны три графа, которые использованы в примере.



а) Граф G'_{SC_1}



б) Граф G'_{SC_2}



в) Граф G'_{SC_3}

Рисунок 2.3 – Примеры графов зависимостей, составляющих конфликтное множество G'_{SC}

На этапе выделения устойчивых словосочетаний по данному предложению было сформировано множество словосочетаний *Term*={современный этап, развитие творческих способностей детей, образовательные учреждения, сетевые научные образовательные сообщества}. Каждое словосочетание представляется графом синтаксических конструкций $G'_{S_i} \in G'_S$ (рис. 2.4).

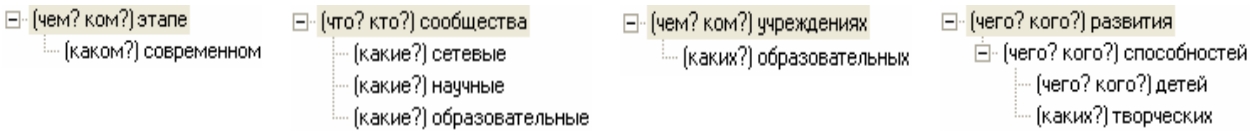
Представим графы как множества:

G'_{S_1} ={этап→современный},

G'_{S_2} ={развитие→способностей, способностей→творческих, способностей→детей},

G'_{S_3} ={учреждения→образовательные},

G'_{S_4} ={сообщества→сетевые, сообщества→научные, сообщества→ образовательные}.



а) Граф G'_{S_1} б) Граф G'_{S_2} в) Граф G'_{S_3} г) Граф G'_{S_4}

Рисунок 2.4 –Графы зависимостей, составляющих конфликтное множество G'_S

Представив графы зависимостей синтаксического анализа аналогичным образом, можно записать условие ядра для анализа рассматриваемого предложения:

$$\left|G'_{SC_j} \cap G'_{S_1}\right| = t' \wedge \left|G'_{SC_j} \cap G'_{S_2}\right| = t' \wedge \left|G'_{SC_j} \cap G'_{S_3}\right| = t' \wedge \left|G'_{SC_j} \cap G'_{S_4}\right| = t' \qquad (2.10)$$

Обозначим мощности пересечения множеств синтаксических конструкций именного субстантивного словосочетания и множеств синтаксических конструкций графа зависимостей как входные лингвистические переменные IS_1 (Intersection), IS_2, IS_3, IS_4. Выражение (2.10) преобразуется в формулу:

$$IS_1=t' \wedge IS_2=t' \wedge IS_3=t' \wedge IS_4=t'. \qquad (2.11)$$

Тогда ядро продукции в общем виде запишется следующим образом:

ЕСЛИ ($IS_1=t' \wedge IS_2=t' \wedge IS_3=t' \wedge IS_4=t'$) ТО $dvh = \Delta t''$. (2.12)

Фрагмент системы нечетких продукционных правил, построенных по формуле 2.8, приведен на рисунке 2.5.

1. If (is1 is big) and (is2 is big) and (is3 is big) and (is4 is big) then (dvh is high) (1)
2. If (is1 is small) and (is2 is big) and (is3 is big) and (is4 is big) then (dvh is down_high) (1)
3. If (is1 is big) and (is2 is big) and (is3 is small) and (is4 is big) then (dvh is down_high) (1)
4. If (is1 is big) and (is2 is small) and (is3 is big) and (is4 is big) then (dvh is up_norm) (1)
5. If (is1 is big) and (is2 is big) and (is3 is big) and (is4 is small) then (dvh is up_norm) (1)
6. If (is1 is small) and (is2 is big) and (is3 is small) and (is4 is big) then (dvh is norm) (1)
7. If (is1 is big) and (is2 is small) and (is3 is small) and (is4 is big) then (dvh is down_norm) (1)
8. If (is1 is small) and (is2 is big) and (is3 is big) and (is4 is small) then (dvh is down_norm) (1)
9. If (is1 is small) and (is2 is small) and (is3 is small) and (is4 is big) then (dvh is up_small) (1)
10. If (is1 is small) and (is2 is big) and (is3 is small) and (is4 is small) then (dvh is up_small) (1)
11. If (is1 is small) and (is2 is small) and (is3 is small) and (is4 is small) then (dvh is small) (1)

Рисунок 2.5 – Фрагмент системы нечетких продукционных правил

Результаты применения системы нечеткого логического вывода с разработанными нечеткими продукционными правилами к графам зависимостей рассматриваемого предложения приведены в таблице 2.5. В

качестве функции принадлежности использована треугольная функция. Дефаззификация выполнена на основе применения метода центра тяжести. Процедура агрегирования нечеткого логического вывода сводится к выбору максимального значения функций истинности:

$$\mu'(\varphi,\psi,\gamma,\vartheta) = \max\{\mu_1(\varphi),\mu_2(\psi),\mu_3(\gamma),\mu_4(\vartheta)\}. \quad (2.13)$$

Таблица 2.5 – Результаты выбора графа зависимостей

Граф	IS_1		IS_2		IS_3		IS_4		dvh
	Pow_1	μ_{IS_1}	Pow_2	μ_{IS_2}	Pow_3	μ_{IS_3}	Pow_4	μ_{IS_4}	
G'_{SC_1}	1	1	3	1	1	1	3	1	0,948
G'_{SC_2}	1	1	2	0,67	1	1	3	1	0,926
G'_{SC_3}	1	1	1	0,33	1	1	3	1	0,622

Итак, из трех графов зависимостей, принадлежащих конфликтному множеству, выбран граф G'_{SC_1} с максимальным значением 0.948.

Таким образом, для реализации синтаксического анализа использованы классические и нечеткие продукции.

2.5 Специальные методы построения онтологий

Онтология описывает понятия определенной предметной области, и, как минимум, включает машиночитаемые определения основных понятий предметной области и отношения между ними. В этом смысле знания становятся возможными для повторного использования людьми, базами данных и приложениями. При этом значительно повышается эффективность как интеллектуальных систем, так и традиционных информационных систем [22]. Этим определяется актуальность создания онтологий. К настоящему времени разработано достаточно много систем, позволяющих в диалоговом режиме создавать онтологии. Однако этот процесс характеризуется высокой трудоемкостью. Поэтому знания о понятиях необходимо извлекать из полнотекстовых источников знаний и автоматически строить онтологии [33,40,41,115]. Так, например, для создания терминосистемы, являющейся ядром онтологии предметной области, знания можно извлечь из терминологических и толковых словарей [106,136]. Проекции терминосистемы на конкретные области знаний (задача, вид деятельности) называются номенклатурами [45]. Для построения номенклатуры знания можно извлекать из научных и учебных изданий.

2.5.1 Построение терминосистемы предметной области

Возможность автоматического построения онтологии предметной области обеспечивается извлечением знаний из качественных терминологических и/или толковых словарей. При этом терминосистема, созданная на основе знаний, извлеченных из словарей, представляет собой

ядро онтологии предметной области. Конечный вариант онтологии должен быть создан посредством соединения нескольких ядер онтологий, построенных на основе разных терминологических словарей. Надо отметить, что терминологические словари, предназначенные для создания онтологии предметной области, должны отбираться экспертом в рассматриваемой области знаний. Для построения полной онтологии предметной области необходимо построить номенклатуры предметной области, которые строятся на основе извлечения знаний из таких научных текстов, как монография, учебное пособие, статья и других. Затем необходимо соединить терминосистему и номенклатуры.

2.5.1.1 Терминологические словари как источники знаний

Существует несколько классификаций (типологий) словарей. Тип любого словаря определяется характером отражаемого лексического материала и практическим значением [45]. Так, энциклопедические (от греч. *enkyklios paideia* – обучение по всему кругу знаний) словари содержат экстралингвистическую информацию об описываемых языковых единицах. Эти словари содержат сведения о научных понятиях, терминах, исторических событиях, персоналиях, географии и т.п. В энциклопедическом словаре нет грамматических сведений о слове, а в зависимости от объёма и адресата словаря даётся более или менее развёрнутая научная информация о предмете, обозначаемом словом. Объектом описания лингвистических (языковых) словарей являются языковые единицы – слова, словоформы, морфемы. В таком словаре слово может быть охарактеризовано с разных сторон в зависимости от целей, объёма и задач словаря: со стороны смыслового содержания, словообразования, орфографии, орфоэпии, правильности употребления.

Кроме этого, различают словари с точки зрения отбора лексики: словари тезаурусного типа и словари, в которых лексика отбирается по определённым параметрам. Например, по сфере употребления различают разговорный, просторечный, диалектный, терминологический словари. По исторической перспективе – словари архаизмов, историзмов, неологизмов и т.п. С точки зрения раскрытия отдельных аспектов (параметров) слова в словарях могут быть этимологическими, грамматическими, орфографическими и т.п. С точки зрения раскрытия системных отношений между словами выделяют гнездовые, словообразовательные, омонимические, паронимические (план выражений), синонимические, антонимические (план содержания) словари.

В данной работе словари рассматриваются с точки зрения возможности их использования в качестве источников знаний. При этом для построения предметной онтологии нужны только качественные словари, содержащие не только определение термина, но и описание свойств, отношений, синонимов и иных элементов знания о термине. Поэтому в качестве источников знаний лучше использовать словари, в которых дается более или менее развёрнутая научная информация о предмете. Такая информация встречается в

энциклопедических, толковых, терминологических и больших предметных словарях.

Любой словарь состоит из словарных статей. Словари отличаются структурированностью словарных статей [69].

Основная часть словарей не имеет четкой структуры словарных статей. Как правило, в словарной статье дается одно или несколько определений (дефиниций) понятий, а затем описываются отношения термина с другими понятиями, посредством которых объясняется суть термина. Эти отношения могут носить родовидовой характер, часть-целое, задавать метрику термина, описывать свойства термина, определять процессы, которые происходят с термином или над термином и т.п.

Словарная статья любого словаря начинается с заглавного слова, являющегося именем термина или терминологического сочетания. Заглавное слово может быть набрано прописными буквами, выделено жирным шрифтом или другим способом. За заглавным словом следует текст, разъясняющий заголовочную единицу в словаре и описывающий её основные характеристики. По степени структурированности данного текста можно выделить словари, имеющие строго определенную структуру словарных статей. При этом нельзя сказать, что эти словари имеют что-то отличительное в своем названии. Один терминологический словарь может иметь четкую структуру словарной статьи, другой – нет. Поэтому рассмотрим просто примеры словарей. Представителем словарей, обладающим четкой структурой словарной статьи, является терминологический словарь по основам информатики и вычислительной техники [34], словарная статья которого имеет четыре части:

– заголовочная часть, содержащая заголовок словарной статьи и дефиницию термина;

– сочетаемостная часть, раскрывающая связи термина или терминологического сочетания с другими словами в предложении, связном тексте;

– иллюстративная часть, демонстрирующая реальное употребление термина или терминологического сочетания в связной речи;

– справочная часть, раскрывающая происхождение термина или терминологического сочетания.

При этом в каждой части словарной статьи можно выделить структурные элементы, которые имеют определенный порядок следования. Так, заголовочная часть начинается с заголовка словарной статьи. Для терминологического сочетания за заголовком указывается его сокращение: например, ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО (ЗУ). В заголовочной части слов-терминов за заголовком следует грамматическая характеристика: приводятся окончание родительного падежа единственного числа, полностью форма множественного числа, окончание родительного падежа множественного числа и указание на род существительного. Далее идет развернутое толкование, раскрывающее структуру понятия термина и его составных частей, приводятся его английское и немецкое соответствия.

Отдельные значения многозначных терминов отмечаются порядковым номером, за которым следуют их толкования. Для примера приведем словарную статью термина «Программирование».

ПРОГРАММИРОВАНИЕ, *-я*; *только ед.*; *ср*. 1. Процесс разработки программы в соответствии с алгоритмом решения задачи, ее отладки и дальнейшего развития программы в ходе ее применения; *англ*, programming; *нем*. Programmierung. 2. Раздел информатики, изучающий методы и приемы построения, отладки и развития программ для ЭВМ; *англ*, programming; *нем*,- Programmierung.

Много внимания в словарной статье уделяется описанию сочетаемости слова-термина или терминологического сочетания. Сведения о сочетаемости помогают осмыслить структуру понятия термина или терминологического сочетания и осознать сущность их основных характеристик. В начале сочетаемостной части словарной статьи приводятся примеры сочетания заголовочного термина с прилагательным: например, п р о г р а м м а автономная, наборная, обучающая, оптимальная, отладочная, параллельная, прикладная, простая, рабочая, разветвляющаяся, системная, сложная, специальная, стандартная, типовая, учебная, целевая, циклическая и т.д. При этом все выражения, с которыми сочетаются слова-термины, приводятся в алфавитном порядке. Далее идут конструкции с приложением: например, п р о г р а м м а -загрузчик, п р о г р а м м а -интерпретатор, п р о г р а м м а -компилятор, п р о г р а м м а -переводчик, п р о г р а м м а -транслятор и т.д. После приводятся сочетания заголовочного слова-термина или терминологического сочетания с другими словами и конструкциями. При этом различаются сочетания, в которых заголовочное слово-термин выступает в качестве опорного слова, и сочетания, в которых заголовочное слово-термин выступает в качестве зависимого слова. Здесь даются вначале беспредложные, затем предложные конструкции. Приведем пример сочетаний, в которых заголовочное слово-термин является опорным словом: В в о д величин, данных, информации, программы, слагаемых, слов, строк, текста, чисел и т.д. Далее следуют сочетания глагола-инфинитива с зависимым словом-термином, терминологическим сочетанием: например, пользоваться, управлять п р о г р а м м о й. Заключают данную часть словарной статьи предикативные сочетания, в которых заголовочное слово-термин является подлежащим, а в качестве сказуемого выступают глаголы в личной форме, страдательные причастия: например, п р о г р а м м а имеет какой-л. вид, содержит какие-л. команды. Во всех словарных статьях рассматриваемые термины в сочетаниях выделены разреженным шрифтом.

В словарных статьях, представляющих собой терминологические гнезда, вслед за сочетаемостью приводятся терминологические сочетания, связанные с заголовочной единицей. Например, в словарной статье с заголовочным термином АЛГОРИТМ приводятся следующие терминологические сочетания и их дефиниции.

Алгоритм вспомогательный – алгоритм, исполнение которого задается путем его вызова из другого алгоритма, основного по отношению к данному алгоритму.

Алгоритм линейный – алгоритм без ветвлений и циклов, выполняющийся строго последовательно, в порядке записи команд.

Алгоритм основной – алгоритм, в котором есть команда вызова другого, вспомогательного алгоритма.

Иллюстративная часть словарной статьи выделена курсивом и содержит примеры употребления терминов и терминологических сочетаний в связной речи. Иллюстративный материал отобран из научных и научно-популярных текстов. И, с одной стороны, он показывает, как используется тот или иной термин в речи. С другой стороны, приведенные примеры позволяют уточнить и расширить представление о содержании термина, обратить внимание на наиболее важные и существенные элементы значения термина, его функциональную нагрузку, показать наиболее типичное употребление термина, терминологического сочетания в связной речи, раскрыть его связи с другими словами в предложении, тексте.

Словарную статью завершает справочная часть, которая содержит сведения о происхождении иноязычного термина и его компонентов и заключена в квадратные скобки: [Файл – от англ. file – досье, картотека.].

Как видно из вышеизложенного, рассмотренный терминологический словарь имеет определенную структуру словарной статьи. Но большинство терминологических словарей представляются словарными статьями, пояснительная часть которых содержит варианты толкования термина, перекрестные ссылки, синонимы и области использования слов. Так, например, в терминологических словарях [91,92] слова-термины представлены в алфавитном порядке. Составные термины (терминологические сочетания) включены в словарь в соответствии с наиболее важными своими составными компонентами, которыми, как правило, являются существительные. Основная словарная статья посвящена современному широко распространенному термину, а все остальные термины, в том числе и его синонимы, содержат перекрестную ссылку на эту статью. Перекрестные ссылки построены в направлении от общего к частному. В некоторых случаях ссылка «*см.* заголовок словарной статьи» или «*см. также* заголовок словарной статьи» от частного к общему используется для того, чтобы информировать о терминах, связанных с данными словарными статьями. Ссылками типа «*ср.* заголовок словарной статьи» в словарных статьях помечены термины, которые обладают сравнительным значением по отношению к отсылаемому термину. Для многих терминов приводится несколько определений, основанных на их употреблении в различных предметных областях. При этом первым приводится наиболее употребительное определение, а остальные толкования даются в соответствии с указанной предметной областью. Например:

Float: колебаться (о курсах валют), свободно плавать (о курсах валют). (1) *банковское дело*: суммы, находящиеся в процессе взимания,

представленные чеками, принадлежащими одному банку, но выписанными на другие банки, местные или расположенные за пределами города. См. также *uncollected funds*. (2) *ценные бумаги*: часть новых ценных бумаг, еще не проданных в открытой продаже. Ср. *undigested securities*.

В некоторых статьях за приведенным определением следует слово «синоним». Но оно не означает, что термин является точным эквивалентом основного заголовка словарной статьи, в которой он приведен. Обычно этот термин лишь приблизительно отражает основной смысл заголовка.

Таким образом, терминологические словари содержат терминологию одной или нескольких специальных областей знаний или деятельности, используемую в современном мире. В них даются основные понятия, без которых трудно обойтись в конкретной деятельности, и довольно подробные объяснения. Структура словарных статей терминологических словарей различна и разрабатывается составителями для каждого словаря. Степень структурированности содержания словарной статьи, порядок следования ее структурных элементов, полнота изложения зависят от целевого назначения словаря, специфики области знаний.

2.5.1.2 Построение семантической сети знаков-фреймов как модели представления терминосистемы

Основным назначением метода извлечения знаний из терминологических словарей является заполнение извлеченной информацией элементов словарных статей тезауруса, структуры которых определены в разделе 1.3.

По сути, графическая интерпретация знака “понятие” t является центральным звеном модели представления знаний и отождествляется с элементарным фрагментом Ф семантической сети SF предметной области (2.14):

$$t \xrightarrow{def} \Phi, \tag{2.14}$$

где Ф – знак-фрейм семантической сети.

Так как каждая вершина такого знака-фрейма представляет собой вектор или множество, то она раскрывается пучком компонентов вектора или множества, которые, в свою очередь, могут тоже раскрываться пучком компонентов. Таким образом, будет построена единая семантическая сеть SF знаков-фреймов.

Построение семантической сети знаков-фреймов [57,71], анализ построенной сети, соединение сетей [59] осуществляется посредством методов, представленных в виде систем продукций.

Вначале активизируются продукции, выявляющие множество заголовочных терминов $T = \{z_i\}$. Заголовочными терминами z_i заполняются фреймы-прототипы концептуального объекта «Понятие», в результате формируется множество экзофреймов $\{\Phi_{Ci} | \, i=1..\,|T|\}$.

Следовательно, на начальном этапе будем иметь множество изолированных фреймов $\{\Phi_{Ci}\}$, которые по мере заполнения слотов фрейма-прототипа должны объединиться в единую сеть:

$$SF = \bigcup_{i=1}^{n} \Phi_{Ci}, \qquad (2.15)$$

где n – количество терминов предметной области; Φ_{Ci} – фреймы, описывающие все концептуальные объекты предметной области.

Таким образом, основной процедурой построения сети является последовательный анализ каждой словарной статьи терминологического словаря, который состоит из следующих процессов распознавания:

– дефиниций;
– квантитативных отношений;
– квалитативных отношений.

Распознавание множества дефиниций термина. При извлечении *дефиниции* возникают следующие ситуации:

1) словарная статья может иметь одну или несколько дефиниций;

2) если в статье наличествует одна дефиниция, то она начинается после первого встретившегося в словарной статье символа '–';

3) если в статье присутствует несколько дефиниций, то они могут нумероваться либо арабскими цифрами, либо буквами латинского алфавита;

4) при нумерации дефиниций после номера может стоять либо символ '.', либо символ ')'.

Таким образом, признаками начала дефиниции являются символ '–' или любая из последовательности символов вида "#.", "#)". Здесь символ '#' обозначает арабскую цифру или латинскую букву. Кроме того, дефиниция всегда расположена в первом предложении словарной статьи.

Это означает, что для выявления дефиниции нужно определить, содержит ли первое предложение словарной статьи указанные признаки. Тогда продукционное правило для распознавания одной дефиниции будет иметь вид:

```
ЕСЛИ
<предложение>     p1   содержит   <признак>      h1          И
<признак>         h1   имеет      <значение>     ["-"]       И
<предложение>     p1   имеет      <индекс>       i1          И
<индекс>          i1   имеет      <значение>     [1]
ТО
<предложение>     p1   имеет      <тип>          ["дефиниция"].
```

Постдействием данной продукции является выделение части предложения от символа "–" до точки.

В случае, когда термин имеет две дефиниции, которые следуют за признаками «1)» и «2)» соответственно, распознавание дефиниций будет осуществляться посредством правил:

Продукционное правило для распознавания первой дефиниции:

```
ЕСЛИ
<предложение>     p1   содержит   <признак>      h1          И
<признак>         h1   имеет      <значение>     ["1)"]      И
<предложение>     p1   имеет      <индекс>       i1          И
<индекс>          i1   имеет      <значение>     [1]
```

```
ТО
<предложение>      p1  имеет <тип>       ["дефиниция"];
```

Продукционное правило для распознавания второй дефиниции:

```
ЕСЛИ
<предложение>      p1  содержит   <признак>        h1          И
<признак>          h1  имеет      <значение>       ["2)"]      И
<предложение>      p1  имеет      <индекс>         i1          И
<индекс>           i1  имеет      <значение>       [1]
ТО
<предложение>      p1  имеет      <тип>            [“дефиниция”].
```

Постдействием данных продукций является выделение частей предложения от «1)» до «2)» и от «2)» до точки.

Рассмотрим пример распознавания дефиниций в словарной статье, содержащей два определения термина «акт»:

«АКТ – 1) официальный документ, имеющий юридическую силу; 2) единичное действие, поступок. В зависимости от статуса органа, его принявшего (выпустившего), акты разделяются на государственные, региональные и ведомственные».

После активизации продукций найденными дефинициями заполняются слоты с атрибутом `NAME="definition"` экзофрейма Φ_{ci}, релевантного рассматриваемому заголовочному термину z_i. Выделенные дефиниции присваиваются атрибуту «VALUE»:

```
<SLOT NAME="definitions">
     <SLOT NAME="definition" VALUE="официальный документ, имеющий
                                    юридическую силу"/>
     <SLOT NAME="definition" VALUE="единичное действие, поступок"/>
</SLOT>.
```

Количество вложенных слотов с атрибутом `NAME="definition"` определяется количеством дефиниций термина.

Распознавание множества квантитативных отношений. Этот процесс включает распознавание синонимов и коррелятов.

Отношение синонимии. Формирование множества терминов, имеющих квантитативные отношения с рассматриваемым термином, рассмотрим на примере формирования множества *синонимов термина*. Наиболее распространенной является ситуация, когда синонимы рассматриваемого термина следуют за терм-признаком «Син.:» или «Синоним:». Но в некоторых словарях синонимы термина не указываются явно, и могут быть выявлены только в результате анализа текста словарной статьи, например:

«Авизо (итал. avviso, англ, advice – сообщение, уведомление) – извещение, посылаемое одним контрагентом другому об изменениях в состоянии взаимных расчетов, о переводе денег, посылке товаров или расчетов с третьими лицами».

Здесь синонимом заголовочного термина является перевод аналога термина на иностранном языке. В данном случае перевод содержится в первом предложении и следует за заголовочным термином после пары терм-признаков «англ.,» и '–', или «итал.,» и '–', или «нем.,» и '–', и т.п.

В системе продукций, предназначенной для распознавания синонимов, должны присутствовать правила, рассматривающие все возможные ситуации определения синонима в словарной статье.

Например, словарная статья термина «акция» содержит предложение:

«Акция – осуществляемое по заранее разработанному плану масштабное действие, син.: деяние, действие».

Тогда продукционное правило, распознающее синоним термина, для случая, когда предложение содержит признак “син.:”, будет записано в виде:

```
ЕСЛИ
<предложение>      p   содержит   <термин>          z          И
<предложение>      p   содержит   <список>          q          И
<список>           q   содержит   <ЭлементСписка>   e          И
<предложение>      p   содержит   <признак>         h          И
<признак>          h   имеет      <значение>        ["син.:"]  И
<признак>          h   имеет      <индекс>          i          И
<список>           q   имеет      <индекс>          (i+1)
ТО
<ЭлементСписка>    e   имеет      <тип>             ["синоним"].
```

Постдействием данной продукции является выделение списка терминов, следующего за указанным в продукции признаком.

После активизации продукции найденные синонимы "деяние" и "действие" выделяются и присваиваются атрибуту «VALUE» соответствующих слотов с NAME="namesyn" экзофрейма Φ_i, релевантного термину «акция»:

```
<SLOT NAME="synonyms">
     <SLOT NAME="synonym">
       <SLOT NAME="namesyn"  VALUE="деяние"/>
       <SLOT NAME="link"     FILE="Frame_#.xml"/>
     </SLOT>
     <SLOT NAME="synonym">
       <SLOT NAME="namesyn"  VALUE="действие"/>
       <SLOT NAME="link"     FILE="Frame_#.xml"/>
     </SLOT>
</SLOT>.
```

Далее производится поиск терминов-синонимов во множестве фреймов Ф. Если экзофреймы Φ_{Ci} и Φ_{Cj}, описывающие данные термины, найдены, то в соответствующих слотах с атрибутом NAME="link" атрибуту FILE="Frame_#.xml" присваиваются ссылки на Φ_{Ci} и Φ_{Cj} (полный путь к файлу, символ “#” в имени файла Frame_#.xml обозначает порядковый номер файла). В противном случае создаются фреймы, релевантные терминам "деяние" и "действие", и после этого ссылки на них записываются в соответствующие слоты.

Распознавание квалитативных отношений. К данной категории семантических отношений относятся группы отношений иерархии и агрегации, функциональные и семиотические отношения. Рассмотрим процесс распознавания квалитативных отношений на примере основных отношений «Объект-свойство», «Целое-часть», «Род-вид», «Субъект действия-действие-объект действия».

Отношение «Объект-свойство». Данное семантическое отношение, определяющее свойства термина, относится к группе отношений агрегации. Под свойствами понимаются различные атрибуты объекта, описывающие различные стороны проявления качеств данного объекта. К таким атрибутам объекта относятся признаки, характеристики, параметры, факторы, критерии.

Анализ текста словарных статей показал, что предложения, описывающие свойства заголовочного термина, обычно содержат такие семантические отношения, как «обладает», «имеет», «характеризуется», кроме того, они определяются наличием терм-спутников «свойство», «признак», «фактор», «критерий», «параметр». В связи с этим для распознавания данного отношения были определены множество терм-спутников; множество глаголов, характеризующих семантическое отношение «Объект-свойство». На основе данных множеств были построены диагностирующие конструкции вида *xRy*, примеры которых приведены ниже:

– *x* “обладает свойством|признаком|параметром|критерием|… ” *y*;
– *x* “имеет свойство|признак|параметр|критерий|…” *y*;
– *x* “характеризуется наличием” *y*;
– “для” *x* “характерно” *y*;
– “свойствами|признаками|параметрами| … ” *x* “являются” $y_1, \ldots, y_n$;
– “к свойствам|признакам|параметрам|…” *x* “относятся:” $y_1, \ldots, y_n$;
– *x* “имеет свойства|признаки|параметры|…” $y_1,\ldots, y_n$;
– *x* “имеет недостатки:” $y_1,\ldots, y_n$;
– “характер” *x* “определяется” $y_1,\ldots, y_n$.

В данных конструкциях в кавычках приведены семантические отношения *R* и терм-спутники терминов *x* и *y*.

Рассмотрим распознавание отношения «Объект-свойство» на примере предложения:

«Характер аграрных отношений определяется господствующим социально-экономическим строем, уровнем развития производительных сил, сложившимися традициями».

Продукционное правило, описывающее данную ситуацию, имеет вид:

```
ЕСЛИ
<предложение>       p   содержит    <термин>          z          И
<предложение>       p   содержит    <список>          q          И
<предложение>       p   содержит    <СемОтношение>    r          И
<предложение>       p   содержит    <терм-спутникX>   tx         И
<СемОтношение>      r   содержит    <глагол>          v          И
<список>            q   содержит    <ЭлементСписка>   e          И
<глагол>            v   имеет       <значение>   ["определяется"] И
<терм-спутникX>     tx  имеет       <значение>   ["характер"]     И
<термин>            z   имеет       <индекс>          i          И
<терм-спутникX>     tx  имеет       <индекс>          (i-1)      И
<СемОтношение>      r   имеет       <индекс>          (i+1)      И
<список>            q   имеет       <индекс>          (i+2)
ТО
<ЭлементСписка>     e   имеет       <тип>       ["свойство"].
```

Аналогичным образом строятся правила для формирования множества свойств термина, описывающие другие возможные ситуации.

Постдействием продукции является выделение свойства термина. Затем выделенные свойства термина z_i записываются в слот «Свойства» экзофрейма Φ_{ci}.

```
<SLOT NAME="propertys">
  <SLOT NAME="property">
       <SLOT NAME="nameprop" VALUE="господствующий
                             социально-экономический строй"/>
       <SLOT NAME="link"     FILE="Frame_#.xml"/>
```

```
    </SLOT>
    <SLOT NAME="property">
        <SLOT NAME="nameprop" VALUE="уровнь развития
                                    производительных сил"/>
        <SLOT NAME="link"      FILE="Frame_#.xml"/>
    </SLOT>
    <SLOT NAME="property">
        <SLOT NAME="nameprop" VALUE="сложившиеся традиции"/>
        <SLOT NAME="link"      FILE="Frame_#.xml"/>
    </SLOT>
</SLOT>.
```

В том случае, если *k*-е свойство p_{ik} является величиной *q* (как, например, уровень производительных сил), то создается экзофрейм с типом концептуального объекта «Величины» Φ_{q1}, и в Φ_{ci} в слот с атрибутом NAME="link" добавляется ссылка на этот экзофрейм. Если свойство p_{ik} является понятием *t*, то происходит анализ на включение его во множество *T*; если такое понятие найдено, и оно релевантно экзофрему Φ_{cj}, то в экзофрейме Φ_{ci} записывается ссылка на Φ_{cj}. Например, свойства «господствующий социально-экономический строй» и «сложившиеся традиции» термина «аграрные отношения» являются, в свою очередь, понятиями. Если понятие *t* не включено во множество *T*, то оно добавляется в это множество, и по нему создается новый экзофрейм, ссылка на который записывается в слоте «Свойства» экзофрейма Φ_{ci}.

Таким образом, распознавание множества свойств термина основано на выявлении семантического отношения *R* и терм-спутников *x* и *y*.

Отношение «Целое-часть». Данное отношение относится к группе отношений агрегации. Основными диагностирующими конструкциями для распознавания отношения «Целое-часть» в тексте словарной статьи являются:

– *x* “имеет часть|компонент|элемент|звено|уровень” *y*;
– *x* “составляет %” *y*;
– *x* “составляет часть” *y*;
– *x* “ – составная часть” *y*;
– *x* “является частью|элементом|звеном|компонентом” *y*;
– *x* “содержит” *y*;
– *x* “содержит части|компоненты|элементы|звенья|уровени:” *y*;
– *x* “включает в себя” *y*;
– *x* “включается в” *y*;
– в *x* “выделяются части|компоненты|элементы|звенья|…” $y_1,\ldots, y_n$;
– в *x* “входят” $y_1,\ldots, y_n$;
– частью|компонентом|элементом *x* “выступает” *y*;
– *x* “представляет собой часть” *y*.
– *x* “представляет собой систему” $y_1,\ldots, y_n$;
– и другие.

Для примера рассмотрим распознавание отношения «Целое-часть» во фрагменте словарной статьи:

«БАЛАНС СЕЛЬСКОГО ХОЗЯЙСТВА – составная часть баланса народного хозяйства. … Составной частью баланса сельского хозяйства

является бухгалтерский баланс предприятия, объединения, кооператива, акционерного общества и других форм хозяйства в аграрном секторе».

Для выделения отношения в первом предложении построим продукции:

```
ЕСЛИ
<предложение>      p   содержит    <термин>          z1          И
<предложение>      p   содержит    <термин>          t1          И
<предложение>      p   содержит    <СемОтношение>    r           И
<СемОтношение>     r   содержит    <глагол>          v           И
<предложение>      p   содержит    <терм-спутникX>   tx          И
<глагол>           v   имеет       <значение>        ["является"]  И
<терм-спутникX>    tx  имеет       <значение>  ["составной частью"]И
<термин>           z1  имеет       <индекс>          i           И
<СемОтношение>     r   имеет       <индекс>          (i+1)       И
<термин>           t1  имеет       <индекс>          (i+2)
ТО
<термин>           z1  имеет       <тип>             ["Целое"]   И
<термин>           t1  имеет       <тип>             ["Часть"]   И
<СемОтношение>     r   относится к <категория>       ["ЦелоеЧасть"].
```

Постдействием продукции является выделение термина t_1, являющегося целым по отношению к заголовочному термину z_1.

Во втором предложении заголовочный термин z_1 выступает целым по отношению к терминам t_i:

```
ЕСЛИ
<предложение>      p   содержит    <термин>          z1          И
<предложение>      p   содержит    <термин>          t1          И
<предложение>      p   содержит    <СемОтношение>    r           И
<СемОтношение>     r   содержит    <глагол>          v           И
<СемОтношение>     r   содержит    <терм-спутникR>   tr          И
<глагол>           v   имеет       <значение>        ["-"]       И
<терм-спутникR>    tr  имеет       <значение> ["составная часть"]И
<термин>           z1  имеет       <индекс>          i           И
<СемОтношение>     r   имеет       <индекс>          (i+1)       И
<термин>           t1  имеет       <индекс>          (i+2)
ТО
<термин>           z1  имеет       <тип>             ["Часть"]   И
<термин>           t1  имеет       <тип>             ["Целое"]   И
<СемОтношение>     r   относится к <категория>       ["ЦелоеЧасть"].
```

Постдействием данной продукции является выделение термина t_1, являющегося частью по отношению к заголовочному термину z_1.

Таким образом, при распознавании семантического отношения «Целое-часть», как это было показано в рассмотренном примере, могут быть найдены множество понятий T_{22}, являющихся частью понятия z_i, и множество понятий T_{21}, являющихся «целым» по отношению к понятию z_i. Имя каждого понятия $t_{21,j} \in T_{21}$ и $t_{22,k} \in T_{22}$ присваивается атрибутам VALUE слота "namewhole", входящего в слот «Whole», и слота "namepart", входящего в слот «Part» соответственно. Слоты «Part» и «Whole», в свою очередь, входят в слот «WholePart», соответствующий семантическому отношению «Целое-часть». Например, понятие z_i «Баланс сельского хозяйства» относится к системе «Баланс народного хозяйства», а его часть является понятием «бухгалтерский баланс предприятия, объединения, кооператива, акционерного общества и других форм хозяйства в аграрном секторе». Тогда в экзофрейме Φ_{ci}, описывающем

термин «Баланс сельского хозяйства», будут заполнены следующие слоты:

```
<SLOT NAME="WholePart">
  <SLOT NAME="Whole">
    <SLOT NAME="namewhole" VALUE="Баланс народного хозяйства"/>
    <SLOT NAME="link" FILE="Frame_#.xml"/>
     <SLOT NAME="attachproc" VALUE="FunWhole"/>
  </SLOT>
  <SLOT NAME="Part">
    <SLOT NAME="namepart"    VALUE="бухгалтерский баланс предприятия,
                                    объединения, кооператива,
                                    акционерного общества и других форм
                                    хозяйства в аграрном секторе"/>
    <SLOT NAME="link"        FILE="Frame_#.xml"/>
    <SLOT NAME="attachproc" VALUE="FunPart"/>
  </SLOT>
</SLOT>
```

Каждое понятие $t_{21,j} \in T_{21}$ и $t_{22,k} \in T_{22}$ должно быть проверено на включение во множество T. Если $\{t_{21,j}\} \cap T \neq \varnothing$ или $\{t_{22,k}\} \cap T \neq \varnothing$, то осуществляется поиск по образцу фрейма, содержащего заголовочный термин z_l и являющегося $t_{21,j}$ или $t_{22,k}$. Пусть фрейм Φ_{cl} содержит заголовочный термин $z_l = t_{21,j}$ = «Экономические отношения», тогда в слоте с атрибутом NAME="link" фрейма Φ_{ci} необходимо указать ссылку на фрейм Φ_{cl}:

<SLOT NAME="link" FILE="Frame_#.xml"/>.

Здесь в имени файла символ ‘#’ означает порядковый номер фрейма в семантической сети знаков-фреймов.

Если $\{t_{21,j}\} \cap T = \varnothing$ или $\{t_{22,k}\} \cap T = \varnothing$, то в этом случае создается новый фрейм для данного термина $t_{21,j}$ или $t_{22,k}$. Во фрейме Φ_{cl} необходимо установить ссылки на этот новый фрейм. При этом возможно, что в результате анализа текущего терминологического словаря остальные слоты этих фреймов останутся незаполненными.

Отношение «Род-вид». Распознавание семантического отношения «Род↔вид» в тексте словарной статьи будем производить на основе выявления семантических отношений R и терм-спутников терминов x и y. К основным диагностирующим конструкциям, используемым при распознавании данного отношения, относятся:

– x “вид –” y;
– x “род –” y;
– x “является видом” y;
– x “является родом” y;
– x “имеет значения:” $y_1,\ldots, y_n$;
– x “могут быть” $y_1,\ldots, y_n$;
– x “различают” $y_1,\ldots, y_n$;
– x “выступают в формах:” $y_1,\ldots, y_n$;
– x “существуют типы|формы|виды:” $y_1,\ldots, y_n$;
– x “подразделяется на группы:” $y_1,\ldots, y_n$;
– x “считается видом” $y_1,\ldots, y_n$;
– и другие.

Рассмотрим процесс распознавания родовидового отношения на примере анализа предложения:

«Баланс платежей – наиболее распространенный вид баланса международных расчетов».

Для выделения отношения в предложении построим продукцию:

```
ЕСЛИ
<предложение>       p   содержит    <термин>          z1          И
<предложение>       p   содержит    <термин>          t1          И
<предложение>       p   содержит    <СемОтношение>    r           И
<СемОтношение>      r   содержит    <глагол>          v           И
<предложение>       p   содержит    <терм-спутникX>   tx          И
<глагол>            v   имеет       <значение>        ["-"]       И
<терм-спутникX>     tx  имеет       <значение>        ["вид"]     И
<термин>            z1  имеет       <индекс>          i           И
<СемОтношение>      r   имеет       <индекс>          (i+1)       И
<термин>            t1  имеет       <индекс>          (i+2)
ТО
<термин>            z1  имеет       <тип>             ["Вид"]     И
<термин>            t1  имеет       <тип>             ["Род"]     И
<СемОтношение>      r   относится к <категория>       ["РодВид"].
```

Постдействием продукции является выделение термина t_1, являющегося родом по отношению к заголовочному термину z_1.

В экзофрейме Φ_{Ci}, релевантном термину z_1="Баланс платежей", заполняется первый элемент слота с атрибутом NAME="Class" – слот с атрибутом NAME="nameclass":

```
<SLOT NAME="Class">
     <SLOT NAME="nameclass"   VALUE="баланс международных расчетов"/>
     <SLOT NAME="link" FILE="Frame_#.xml"/>
</SLOT>.
```

Далее производится поиск по образцу термина "баланс международных расчетов" во множестве Φ_C. Если такой экзофрейм Φ_{Cj} найден, то ссылка на него записывается в слот:

```
     <SLOT NAME="link" FILE="Frame_#.xml"/>.
```

Иначе создается экзофрейм Φ_{Cj} с типом концептуального объекта «Понятие», в котором заполняется слот:

<NAMETERM NAMETERM_ID="баланс международных расчетов"/> .

После чего ссылка на созданный фрейм заносится в соответствующий слот экзофрейма Φ_{Ci} термина "Баланс платежей".

В экзофрейме Φ_{Cj} термина "баланс международных расчетов" заполняется слот, описывающий термин "Баланс платежей", являющийся его видом, и прописываются соответствующие ссылки:

```
<SLOT NAME="Kind">
     <SLOT NAME="namekind"    VALUE="баланс платежей"/>
     <SLOT NAME="link" FILE="Frame_#.xml"/>
</SLOT>.
```

При распознавании семантического отношения «Род-вид» могут быть найдены множество понятий T_{11}, являющихся родом z_i, и множество понятий T_{12}, являющихся «видом» по отношению к понятию z_i. В этом случае распознавание родовидового отношения, выделение терминов, создание экзофреймов и их связывание в семантическую сеть осуществляются аналогично построению сети знаков-фреймов для отношения «Целое-часть».

Таким образом, для выявления родовидовых отношений, как и в случае распознавания отношения «Целое-часть», необходимо распознать соответствующие семантические отношения и терм-спутники.

Отношение «Объект действия ↔ действие ↔ субъект действия». Для распознавания в тексте данного отношения будем использовать множество модифицированных графов зависимостей $G'=\{g'\}$, построенных при пофразном синтаксическом анализе текста словарной статьи терминологического словаря. Напомним, что модифицированные графы зависимостей отличаются от обычных графов зависимостей тем, что их вершины, содержащие лексемы, составляющие словосочетания, объединены в единую вершину, которой релевантно это словосочетание. Например, модифицированный граф зависимостей g' предложения «Банковские операции финансируются из обычных источников капитала и специальных средств» имеет вид, показанный на рисунке 2.6.

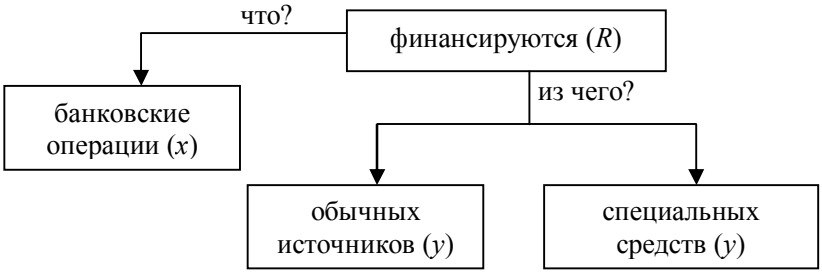


Рисунок 2.6 – Пример модифицированного графа зависимостей

По графу зависимостей предложения можно построить графы синтаксических конструкций g'_{S_i} в виде пары вершин первого уровня и отношения зависимости между ними вида «глагол *v* – термин *t*». Например, для графа, приведенного на рисунке 2.6, графы синтаксических конструкций g'_{S_i} можно представить как множества вида:

g'_{S_1} = {финансируются → банковские операции},

g'_{S_2} ={финансируются→обычных источников},

g'_{S_3} ={финансируются→специальных средств}.

Представив графы зависимостей подобным образом и проведя анализ полученных графов синтаксических конструкций g'_{S_i}, можно распознать в предложении отношение «Объект действия-действие-субъект действия». Известно, что в русском языке отношение действия к субъекту (производителю действия) и объекту действия (предмету, над которым действие производится) выражается глагольной категорией, которая называется грамматической категорией залога [18]. Основными являются действительный и страдательный залоги. Действительный залог имеют глаголы, обозначающие действие, производимое субъектом и активно направленное на объект. Действительный залог имеет синтаксическую характеристику: субъект действия является подлежащим, а объект – дополнением в винительном падеже без предлога. Страдательный залог по

значению представляет действие, пассивно направленное от объекта к субъекту. Важнейшим грамматическим показателем страдательного залога является творительный падеж существительного со значением реального субъекта действия.

Таким образом, были определены следующие схемы для распознавания отношения «Объект действия-действие-субъект действия»:

1) если синтаксическая конструкция g'_{S_i} соответствует пассивной схеме «глагол страдательного залога $\rightarrow$ термин в винительном падеже без предлога», то в данной конструкции глагол *v* обозначает действие *a*, а термин *t* – объект действия;

2) если синтаксическая конструкция g'_{S_i} соответствует пассивной схеме «глагол страдательного залога $\rightarrow$ термин в творительном падеже», то в данной конструкции глагол *v* обозначает действие *a*, а термин *t* – субъект действия;

3) если синтаксическая конструкция g'_{S_i} соответствует активной схеме «глагол действительного залога $\rightarrow$ термин в именительном падеже», то глагол активного действия *v* обозначает действие *a*, а термин *t* – субъект действия;

4) если синтаксическая конструкция g'_{S_i} соответствует активной схеме «глагол действительного залога $\rightarrow$ термин в винительном падеже без предлога», то глагол активного действия *v* обозначает действие *a*, а термин *t* – объект действия.

Продемонстрируем работу данных схем на примерах.

Рассмотрим предложение «Авизо используется любыми банками для уведомления своих клиентов о движении денежных сумм по их счетам», которое построено по пассивной схеме, то есть содержит глагол страдательного залога. Модифицированный граф зависимостей *g*' данного предложения имеет вид, показанный на рисунке 2.7.

Построим графы синтаксических конструкций g'_{S_i}:

g'_{S_1} = {используется $\rightarrow$авизо},

g'_{S_2} ={используется$\rightarrow$уведомления},

g'_{S_3} ={используется $\rightarrow$любыми банками}.

Граф g'_{S_1} соответствует пассивной схеме «глагол страдательного залога $\rightarrow$ термин в винительном падеже без предлога», следовательно, в данной конструкции глагол «`используется`» обозначает действие, а термин «`авизо`» – объект действия.

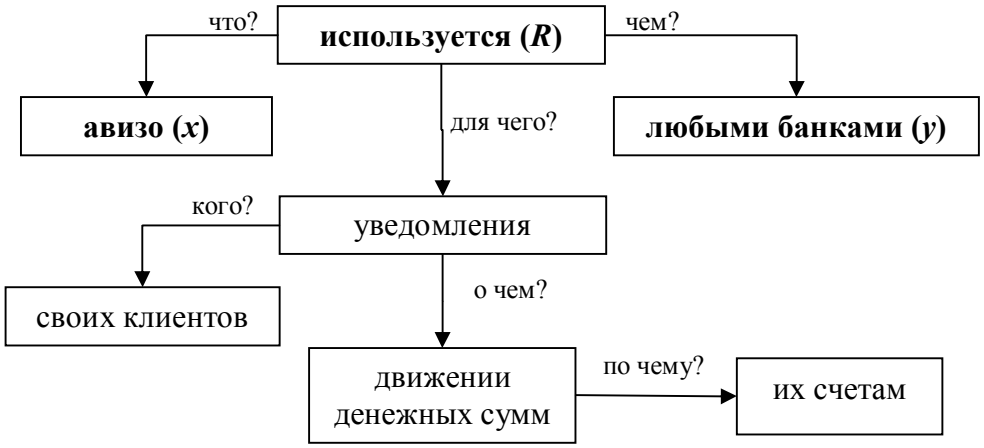


Рисунок 2.7 – Модифицированный граф зависимостей предложения, построенного по пассивной схеме

Таким образом, в результате анализа графа синтаксической конструкции g'_{S_1} заполняется слот экзофрейма Φ_i термина «авизо» с типом концептуального объекта «Понятие»:

```
<SLOT NAME="action">
     <SLOT NAME="nameact"     VALUE="используется"/>
     <SLOT NAME="link" FILE="Frame_#.xml"/>
</SLOT>.
```

После этого необходимо выполнить поиск по образцу глагола «используется» во множестве Ф. Если такой экзофрейм Φ_j найден, то необходимо записать ссылку на него в слоте:

```
<SLOT NAME="link" FILE="Frame_#.xml"/>.
```

Если такой экзофрейм отсутствует, то создается экзофрейм Φ_j с типом концептуального объекта «Действие», в котором заполняется слот:

```
<NAMETERM NAMETERM_ID="используется"/> .
```

После чего ссылка на созданный фрейм заносится в соответствующий слот экзофрейма Φ_i термина «авизо».

Далее производится анализ остальных графов синтаксических конструкций. Граф g'_{S_2} не подходит ни под одну из определенных схем, поэтому из дальнейшего анализа исключается. Граф g'_{S_3} соответствует пассивной схеме «глагол страдательного залога → термин в творительном падеже», то есть в данной конструкции глагол «используется» обозначает действие, а термин «любыми банками» – субъект действия. В результате заполняется следующий слот экзофрейм Φ_j термина «используется»:

```
<SLOT NAME="SubjectObject">
    <SLOT NAME="typeSO" VALUE="субъект"/>
    <SLOT NAME="nameact" VALUE="любыми банками"/>
    <SLOT NAME="link" FILE="Frame_#.xml"/>
</SLOT>.
```

Ссылка на экзофрейм, содержащий описание термина «любыми банками», записывается в слот, если поиск по образцу данного термина во множестве Ф дал положительный результат. В противном случае создается

экзофрейм Φ_k с типом концептуального объекта «Понятие», слот которого заполняется следующим образом:

```
<NAMETERM NAMETERM_ID="банками"/>.
```

Ссылка на созданный экзофрейм Φ_k записывается в соответствующий слот экзофрейм Φ_j термина «используется».

В итоге анализа графа зависимостей предложения, приведенного в примере, заполняется три экзофрейма, описывающих понятия, находящиеся в отношении «Объект действия-действие-субъект действия».

Не все предложения словарной статьи имеют полный состав компонентов. Нередки ситуации, когда в тексте словарной статьи в предложениях, следующих за дефиницией, заголовочный термин, обозначающий субъект действия, опускается. Для примера рассмотрим фрагмент словарной статьи термина «акционер»:

«Акционер (x) (англ. jointstock company) - участник акционерного общества, владелец акций. Получает (v) прибыль (y) в виде дивидендов».

Второе предложение построено по однокомпонентной схеме, в которой отсутствует подлежащее (x) – субъект действия. Глагол «получает» включается во множество действий термина «акционер», дополнение «прибыль» классифицируется как объект действия.

При синтаксическом анализе и построении графов зависимостей предложения данного типа будут преобразованы в двухкомпонентные схемы. И процесс распознавания отношения «Объект действия ↔ действие ↔ субъект действия» сведется к анализу подграфов синтаксических конструкций графа зависимостей с учетом морфологической информации терминов, составляющих данные конструкции.

Интерес представляют предложения вида: Банк России имеет право эмитировать наличные деньги.

Конструкции данного типа содержат составные глагольные сказуемые, которые выражаются личной формой глагола и примыкающим к нему инфинитивом. Глаголы в неопределенной форме составят множество действий термина z_i. Для выявления действий термина можно использовать такие конструкции, как, например:

– z “выполняет функции:” $y_1 t_1,\ldots, y_n\ t_n$;

– z “имеет право” yt;

– z “позволяет” $y_1 t_1,\ldots, y_n\ t_n$;

– z “позволяет” yt ;

– и другие.

Здесь z – заголовочный термин – субъект действия, y_i – действия термина z, t_i – объекты действия.

Рассмотрим пример продукционного правила для анализа предложения данного вида:

```
ЕСЛИ
<предложение>      p   содержит   <термин>          z1          И
<предложение>      p   содержит   <термин>          t1          И
<предложение>      p   содержит   <глагол>          v1          И
<предложение>      p   содержит   <СемОтношение>    r           И
```

```
<СемОтношение>     r    содержит    <глагол>          v2          И
<СемОтношение>     r    содержит    <терм-спутникR>   r           И
<глагол>           v    имеет       <значение>        ["имеет"]   И
<терм-спутникR>    r    имеет       <значение>        ["право"]   И
<список>           q1   содержит    <ЭлементСписка>   e           И
<глагол>           v1   имеет       <характеристика>  x     И
<характеристика>   x    имеет       <значение>        ["инфинитив"]И
<термин>           z1   имеет       <индекс>          i           И
<СемОтношение>     r    имеет       <индекс>          (i+1)       И
<глагол>           v1   имеет       <индекс>          (i+2)       И
<термин>           t1   имеет       <индекс>          (i+3)
ТО
<термин>           z1   имеет       <тип>             ["Субъект"] И
<термин>           t1   имеет       <тип>             ["Объект"]  И
<глагол>           v1   имеет       <тип>             ["Действие"] И
<СемОтношение>     r    относится к <категория>       ["СубДействОб"].
```

В результате активизации продукции выделяются элементы предложения и заполняются соответствующие экзофреймы семантической сети. Процесс создания фреймов и связывания их в сеть аналогичен рассмотренному выше примеру.

Таким образом, при распознавании отношения «Объект действия ↔ действие ↔ субъект действия» необходимо анализировать морфологическую информацию глагола, а также множество терм-спутников (например, право, влияние) как глаголов, так и терминов (например, функции).

2.5.2 Построение номенклатуры предметной области

Извлечение знаний из монологических научных текстов позволяет создать номенклатуру предметной области как проекцию терминосистемы на определенную в научном тексте подобласть знаний или задачу, релевантную предметной области. В этом случае онтология предметной области представляется в виде иерархии, в корне которой находится терминосистема, а в узлах – номенклатура. В качестве модели представления знаний номенклатуры также используется семантическая сеть знаков-фреймов.

Основной подход к построению семантической сети знаков-фреймов номенклатуры тот же, что был использован при построении сети терминосистемы. Семантическая сеть S строится как объединение знаков-фреймов Φ_i (2.16).

$$S = \bigcup_i \Phi_i \ . \qquad (2.16)$$

Будем считать, что посредством метода выделения словосочетаний сформировано множество терминов *Term* (формула 2.1). Это означает, что мощность множества $\Phi=\{\Phi_i\}$ больше мощности множества *Term*.

В научном тексте термин обычно рассматривается с разных точек зрения, в результате чего создается некоторая семантическая окрестность использования термина. В этой окрестности между терминами наблюдается родовидовая связь, для которой может быть построен фрагмент семантической сети. Например, если текст посвящен исследованию проблем создания сетевого сообщества, то могут быть найдены различные виды сообществ: научное сетевое сообщество, его подвиды – сетевое учебно-

методическое образовательное сообщество, сетевые образовательные сообщества по специальностям ВПО и другие. Графическое изображение окрестности термина «Сообщество» из приведенного примера показано на рисунке 2.8.

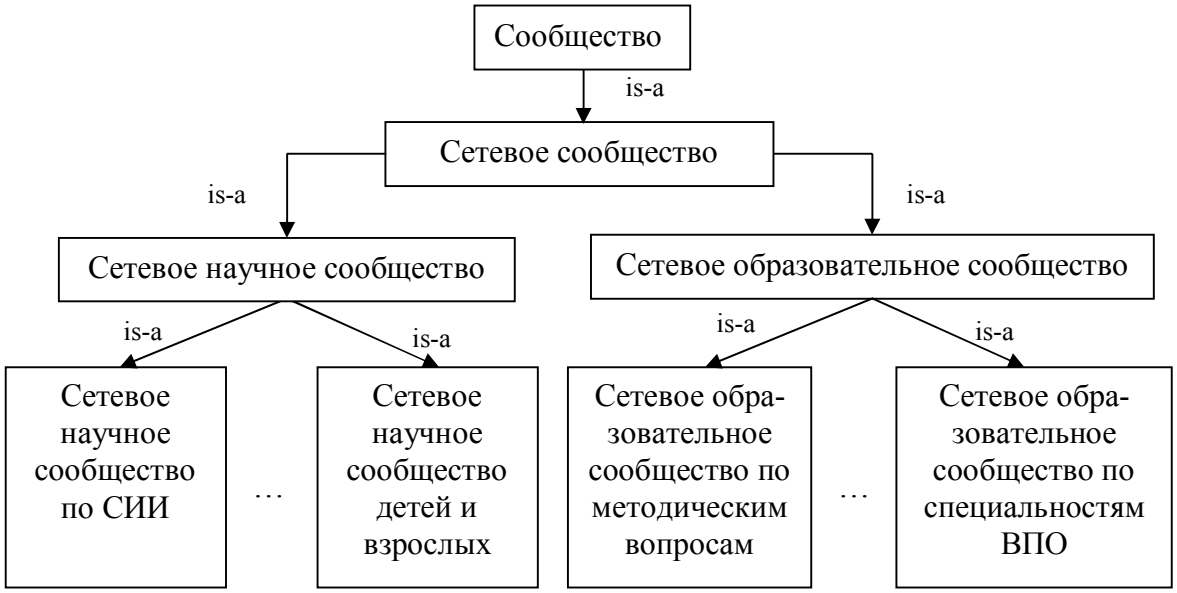


Рис. 2.8 – Фрагмент семантической сети родственных понятий – граф g^*_i

В этом же тексте для выявления видов сообществ могут быть найдены виды деятельностей (научная, образовательная) и их подвиды, виды и подвиды участников сообществ. Таким образом, определив графы семантических окрестностей понятий – множество графов $G^*=\{g^*_i|\ i=1\div m,\ m=|G^*|\}$, мы определяем категории понятий, использованные в научном тексте.

Граф g^*_i является взвешенным и определяется кортежем:

$$g_i^* = (N_i, Ar_i, V_i), \tag{2.17}$$

где N_i – множество вершин графа $N_i=\{t_{ij}|\ t_{ij}\in Term,\ j=1\div r,\ r$ – количество вершин в графе, $|N_i|$ возможно равна 1$\}$, Ar_i – множество дуг в графе, V_i – множество весов вершин графа (терминов). Общий вес графа g^*_i есть аддитивная величина:

$$W_i = \sum_{j=1}^{r} V_{ij}\ . \tag{2.18}$$

Построение семантической сети знаков-фреймов начинается с построения центрального знака-фрейма для термина t^*, находящегося в корне графа g^*_i и имеющего максимальный вес W_i.

Извлечение множества дефиниций термина. Среди типов дефиниций наиболее интересны субстанциональная и тезаурусная (структурная) дифиниции. Субстанциональная дефиниция высвечивает смысл каждого термина, основные функционально важные связи и отношения с другими терминами. Структурная дефиниция по своей функции аналогична не прибору, через который мы видим смысл термина, а экрану, на котором даны валентности каждого термина, отражающие его способность вступать в связи

или отношения с другими терминами, и косвенно отражена каждая из возможных сетей как целостная схема, как та или иная *структура* поля смыслов.

В научном тексте субстанциональная дефиниция может быть дана явно через определение или толкование термина. К основным семантическим отношениям, определяющим субстанциональную дефиницию, относятся следующие глаголы и глагольные группы, являющиеся детекторами этих отношений: “называ[ет] (ем, ется, ют и др.)”, “определяется”, “имеет определение”, “толкуется”, “имеет толкование”, “–это”, “–“. Например, предложение: «Агент – это официальный представитель фирмы, выполняющий посреднические функции», является субстанциональной дефиницией термина «Агент». Для распознавания дефиниции в предложении такого вида используется следующая продукция:

```
ЕСЛИ
<предложение>     p     содержит    <термин>          t     И
<предложение>     p     содержит    <СемОтношение>    r     И
<СемОтношение>    r     содержит    <глагол>          v     И
<глагол>          v     имеет       <значение>        [“-это”]
ТО
<предложение>     p     содержит    <дефиниция>       D.
```

Аналогичным образом строятся продукции для всех глаголов семантического отношения. Постдействием таких продукций будет выделение части предложения, начинающейся после семантического отношения до точки.

Выделенная дефиниция присваивается атрибуту «VALUE» в слоте экзофрейма Φ_1:

```
<SLOT NAME="definition" VALUE="официальный представитель фирмы,
                               выполняющий посреднические функции"/>.
```

В научном тексте, как правило, встречается только одна дефиниция термина. Однако в тексте может отсутствовать толкование термина, в этом случае при необходимости после построения сети субстанциональная дефиниция может быть сформирована исходя из знаний, заложенных во фрейме данного термина.

Таким образом, основным детектором, с помощью которого диагностируется дефиниция термина, является семантическое отношение.

Извлечение множества квантитативных отношений. К этому множеству относятся отношения тождества (синонимии) и оппозиции (корреляции).

Отношение синонимии – это языковое отношение различия между тождественными элементами реальности, которые диагностируются посредством глаголов или глагольных групп семантических отношений: “эквивалентен”, “тождественен”, “имеет синоним”, “является синонимом”, “называется по-разному” и др. Например, в предложении «В зависимости от того, кем является работодатель, или в зависимости от характера оказываемых услуг ЗАРАБОТНАЯ ПЛАТА называется по-разному: жалованье, денежное содержание, должностной оклад, заработок».

В этом случае пример продукции по распознаванию синонима выглядит следующим образом:

```
ЕСЛИ
<предложение>       p   содержит    <термин>          t           И
<предложение>       p   содержит    <список>          q           И
<предложение>       p   содержит    <СемОтношение>    R           И
<предложение>       p   содержит    <признак>         h           И
<СемОтношение>      r   содержит    <глагол>          v           И
<СемОтношение>      r   содержит    <терм-спутникR>   tr          И
<список>            q   содержит    <ЭлементСписка>   e           И
<глагол>            v   имеет       <значение>        [“называется”] И
<терм-спутникR>     tr  имеет       <значение>        [“по-разному”] И
<признак>           h   имеет       <значение>        [‘:’]       И
<термин>            t   имеет       <индекс>          i           И
<СемОтношение>      r   имеет       <индекс>          (i+1)       И
<признак>           h   имеет       <индекс>          (i+2)       И
<список>            q   имеет       <индекс>          (i+3)
ТО
<ЭлементСписка>     e   имеет       <тип>             [“синоним”].
```

Аналогично строятся остальные продукции. Постдействием таких продукций является выделение списка терминов от признака ':' до точки. По каждому термину-синониму списка создается слот:

```
<SLOT NAME="synonym">
      <SLOT NAME="namesyn"    VALUE="?"/>
      <SLOT NAME="link"       FILE="Frame_#.xml"/>
</SLOT>.
```

Заполним вложенные слоты. Атрибуту VALUE присваивается значение элемента списка e_i, например, «жалованье»:

```
<SLOT NAME="namesyn"    VALUE="жалованье"/>.
```

После этого необходимо выполнить поиск по образцу элемента списка e_i во множестве Ф. Если такой знак-фрейм $Ф_j$ найден, то необходимо записать ссылку на него в слоте:

```
<SLOT NAME="link"   FILE="Frame_#.xml"/>.
```

Подобным образом заполняются слоты для остальных синонимов списка q.

Отношение корреляции выражает различия между понятиями и, соответственно, между объектами научного знания.

Корреляты – это термины, относящиеся к одной категории, но противопоставленные по некоторому существенному признаку.

Детекторами диагностики отношения противоположности, противоречия или противопоставления могут являться как семантические отношения (противопоставлен, противоположен, противоречив), терм-спутники глагола семантического отношения (имеет *противоречие*, есть *противоположность*, является *противоположностью* и т.д.), так и сами термины (противоположный процесс, противоположное явление, коррелят и другие). Примерами предложений, содержащих корреляты, являются:

1. «Инфляция – процесс обесценивания денег, в результате которого на одинаковую сумму денег через некоторое время можно купить меньший объем товаров и услуг. На практике это выражается в увеличении цен. Противоположным процессом инфляции является дефляция – снижение цен.

В современной экономике встречается редко и краткосрочно, обычно носит сезонный характер».

2. «В пространстве верх и низ являются коррелятами».

Приведем пример простого продукционного правила.

ЕСЛИ

<предложение>	*p*	содержит	<термин>	t_1	И
<предложение>	*p*	содержит	<термин>	t_2	И
<предложение>	*p*	содержит	<СемОтношение>	*R*	И
<предложение>	*p*	содержит	<признак>	*h*	И
<СемОтношение>	*r*	содержит	<глагол>	*v*	И
<СемОтношение>	*r*	содержит	<терм-спутникR>	*tr*	И
<глагол>	*v*	имеет	<значение>	[“являются”]	И
<терм-спутникR>	*tr*	имеет	<значение>	[“коррелятами”]	И
<признак>	*h*	имеет	<значение>	[‘и’]	И
<термин>	t_1	имеет	<индекс>	*i*	И
<признак>	*h*	имеет	<индекс>	(*i*+1)	И
<термин>	t_2	имеет	<индекс>	(*i*+2)	И
<СемОтношение>	*r*	имеет	<индекс>	(*i*+3)	

ТО

<термин>	t_2	имеет	<тип>	[“коррелят”].

Аналогично строятся остальные продукции. Постдействием продукций является заполнение слота:

```
<SLOT NAME="correlate">
       <SLOT NAME=" "   VALUE="…."/>
       <SLOT NAME="link"      FILE="Frame_#.xml"/>
</SLOT>
```

Заполним вложенные слоты на примере второго предложения. Атрибуту VALUE присваивается значение термина t_2 «низ»:

```
<SLOT NAME=" namecorr"  VALUE="низ"/>.
```

После этого необходимо выполнить поиск по образцу термина t_2 во множестве Ф. Если такой знак-фрейм $Ф_j$ найден, то необходимо записать ссылку на него в слоте:

```
<SLOT NAME="link"       FILE="Frame_#.xml"/>.
```

Извлечение множества квалитативных отношений. Рассмотрим основные из них: «Объект-свойство/признак», «Целое-часть», «Род-вид» и «Объект действия-действие-субъект действия».

Отношение «Объект-свойство/признак». Семантическое отношение «Объект↔свойство/признак», определяющее свойства термина, относится к группе отношений агрегации. Основные диагностирующие выражения для его поиска следующие:

- *x* “обладает свойством” *y*;
- *x* зависит от факторов (показателей|признаков|свойств) *y*;
- *x* “имеет существенным признаком” *y*;
- *x* “–“ *y* (*y* -прилагательное или наречие);
- *x* “характеризуется наличием” *y*;
- *y* “характерно” *x*.

В кавычках указаны семантические отношения вида *R*. Рассмотрим пример предложения:

«Качество труда зависит от следующих факторов: физического здоровья, образования, организаторских способностей, производственного опыта».

Для распознавания отношения «Объект↔свойство/признак» в этом предложении построим продукцию.

```
ЕСЛИ
<предложение>       p   содержит    <термин>          t           И
<предложение>       p   содержит    <список>          q           И
<предложение>       p   содержит    <СемОтношение>    R           И
<предложение>       p   содержит    <признак>         h           И
<СемОтношение>      r   содержит    <глагол>          v           И
<СемОтношение>      r   содержит    <терм-спутникR>   tr          И
<список>            q   содержит    <ЭлементСписка>   e           И
<глагол>            v   имеет       <значение>        ["зависит"] И
<терм-спутникR>     tr  имеет       <значение>        ["факторов"] И
<признак>           h   имеет       <значение>        [':']       И
<термин>            t   имеет       <индекс>          i           И
<СемОтношение>      r   имеет       <индекс>          (i+1)       И
<признак>           h   имеет       <индекс>          (i+2)       И
<список>            q   имеет       <индекс>          (i+3)
ТО
<ЭлементСписка>     e   имеет       <тип>             ["свойство"].
```

Постдействием продукции является выделение списка терминов от признака ':' до точки. По каждому элементу списка создается слот:

```
<SLOT NAME="property">
       <SLOT NAME="nameprop"    VALUE="?"/>
       <SLOT NAME="link"            FILE="Frame_#.xml"/>
</SLOT>.
```

Заполним вложенные слоты. Атрибуту VALUE присваивается значение элемента списка e_i, например, «образование»:

```
<SLOT NAME="namesyn"    VALUE="образование"/>.
```

После этого необходимо выполнить поиск по образцу элемента списка e_i во множестве Ф. Если такой знак-фрейм $Ф_j$ найден, то необходимо записать ссылку на него в слоте

```
<SLOT NAME="link"       FILE="Frame_#.xml"/>.
```

Подобным образом заполняются слоты для остальных элементов списка q.

Отношение «Целое-часть». Данное отношение касается структурированных объектов и показывает, что один объект имеет в своем составе второй объект. Основными диагностирующими конструкциями для распознавания отношения «Целое-часть» являются:

– x “имеет часть|компонент|элемент|звено|уровень” y;
– x “составляет %” y;
– x “составляет часть” y;
– x “является частью|элементом|звеном|компонентом” y;
– x “содержит” y;
– x “содержит части|компоненты|элементы|звенья|уровни:” y;
– x “включает в себя” y;
– x “включается в” y;
– в x “выделяются части|компоненты|элементы|звенья|уровни” y;
– в x “входят” y;
– частью|компонентом|элементом x “выступает” y;
– x “представляет собой часть” y.
– x “представляет собой систему элементов” y;

– и другие.

Для примера рассмотрим распознавание отношения «Целое-часть» в предложении «Аванс составляет 10-40% общей стоимости договора». Для этого построим продукцию:

```
ЕСЛИ
<предложение>       p   содержит    <термин>          t1          И
<предложение>       p   содержит    <термин>          t2          И
<предложение>       p   содержит    <СемОтношение>    R           И
<СемОтношение>      r   содержит    <глагол>          v           И
<СемОтношение>      r   содержит    <терм-спутникR>   tr          И
<глагол>            v   имеет       <значение>  ["составляет"]   И
<терм-спутникR>     tr  имеет       <значение>        ["%"]       И
<термин>            t   имеет       <индекс>          i           И
<СемОтношение>      r   имеет       <индекс>          (i+1)       И
<признак>           h   имеет       <индекс>          (i+2)       И
<термин>            t2  имеет       <индекс>          (i+3)
ТО
<термин>            t1  имеет       <тип>             ["Часть"]   И
<термин>            t2  имеет       <тип>             ["Целое"]   И
<СемОтношение>      r   относится к <категория>       ["ЦелоеЧасть"].
```

Постдействием продукции является последовательное выделение терминов t_1 и t_2. Для каждого термина осуществляется поиск по образцу во множестве Ф. Если такой знак-фрейм Φ_i найден для термина t_1, то в слоте с атрибутом `NAME="Whole"` заполняются вложенные слоты.

В слоте с атрибутом `NAME="namewhole"` атрибуту `VALUE` присваивается значение второго термина t_2, являющегося целым по отношению к термину t_1, а в слоте с атрибутом `NAME="link"`записывается ссылка на экзофрейм Φ_j:

```
<SLOT NAME="Whole">
    <SLOT NAME="namewhole"    VALUE="общей стоимости договора"/>
    <SLOT NAME="link"         FILE="Frame_#.xml"/>
</SLOT>.
```

Ссылка на фрейм определяется в результате поиска терминов t_2 по образцу во множестве Ф. В случае отрицательного результата поиска создается экзофрейм с именем второго термина t_2, и в слот термина записывается ссылка на созданный фрейм.

Отношение «Род-вид». Распознавание семантического отношения «Род-вид» в тексте будем производить на основе анализа графа семантической окрестности g_i^* рассматриваемого текста. Термины t_i, находящиеся в вершинах первого уровня графа g_i^*, составят множество видов термина t^*, находящегося в корне этого графа. В экзофрейме Φ_1, созданном для термина t^*, значениями терминов t_i заполняются слоты с атрибутом `NAME="Kind"`. Далее осуществляется поиск терминов t_i по образцу во множестве Ф; если такие экзофреймы найдены, то в соответствующие слоты записываются ссылки на них, в противном случае для каждого термина t_i создается экзофрейм Φ_i, на который и устанавливается ссылка в соответствующем слоте экзофрейма Φ_1.

Далее для каждого термина t_i в соответствующем экзофрейме Φ_i слоты с атрибутом `NAME="Kind"` заполняются значениями терминов, находящихся в вершинах второго уровня графа g_i^* и являющихся видами термина t_i. Для

данных терминов создаются новые экзофремы, ссылки на которые записываются в соответствующие слоты экзофрейма Φ_i. Этот процесс продолжается до тех пор, пока не будут рассмотрены все вершины графа g_i^*. В конечном итоге в результате анализа графа g_i^* будет построено множество экзофреймов $\Phi=\{\Phi_i\}$, содержащих описание терминов, находящихся в вершинах графа g_i^* и связанных родовидовыми отношениями.

Рассмотрим этот процесс на примере анализа графа g_i^*, изображенного на рисунке 2.8. Для термина «сообщество» видом является термин «сетевое сообщество». В экзофрейме Φ_i заполняется первый элемент слота с атрибутом NAME="Kind" – слот с атрибутом NAME="namekind":

```
<SLOT NAME="Kind">
      <SLOT NAME="namekind"   VALUE="сетевое сообщество"/>
      <SLOT NAME="link"       FILE="Frame_#.xml"/>
</SLOT>.
```

Далее производится поиск по образцу термина «сетевое сообщество» во множестве Ф. Если такой экзофрейм Φ_j найден, то ссылка на него записывается в слот:

```
<SLOT NAME="link"       FILE="Frame_#.xml"/>.
```

Иначе создается экзофрейм Φ_j с типом концептуального объекта «Понятие», в котором заполняется слот:

```
<NAMETERM NAMETERM_ID="сетевое сообщество"/>.
```

После этого ссылка на созданный фрейм заносится в соответствующий слот экзофрейма Φ_i термина «сообщество».

В экзофрейме Φ_j термина «сетевое сообщество» заполняется слот, описывающий термин «сообщество», являющийся его родом:

```
<SLOT NAME="Class">
      <SLOT NAME="nameclass"  VALUE="сообщество"/>
      <SLOT NAME="link"       FILE="Frame_#.xml"/>
</SLOT>.
```

Далее рассматриваются вершины второго уровня графа, содержащие термины «сетевое научное сообщество» и «сетевое образовательное сообщество». В экзофрейме Φ_j термина «сетевое сообщество» в слоте с атрибутом NAME="ClassKind" создаются и заполняются два слота с атрибутами NAME="Kind", описывающие данные термины:

```
<SLOT NAME="Kind">
      <SLOT NAME="namekind"   VALUE="сетевое научное
                                       сообщество"/>
      <SLOT NAME="link"       FILE="Frame_#.xml"/>
</SLOT>
<SLOT NAME="Kind">
      <SLOT NAME="namekind"   VALUE="сетевое образовательное
                                          сообщество"/>
      <SLOT NAME="link"       FILE="Frame_#.xml"/>
</SLOT>.
```

Слоты с атрибутами NAME="link" заполняются ссылками на экзофреймы, описывающие термины «сетевое научное сообщество» и «сетевое образовательное сообщество», найденными в результате поиска во множестве Ф. Если такие экзофреймы во множестве Ф отсутствуют, то

создаются два экзофрейма, в которые записываются знания о данных терминах. В экзофрейме Φ_j термина «сетевое сообщество» указываются ссылки на созданные фреймы. А в созданных фреймах, в свою очередь, создается слот с атрибутом NAME="Class":

```
<SLOT NAME="Class">
     <SLOT NAME="nameclass"   VALUE="сетевое сообщество"/>
     <SLOT NAME="link"        FILE="Frame_#.xml"/>
</SLOT>.
```

Слоты с атрибутом NAME="Kind" создаются и заполняются по результатам анализа вершин третьего уровня рассматриваемого графа семантической окрестности термина «сообщество».

Процесс продолжается до тех пор, пока не будут рассмотрены все вершины данного графа. В результате будет построено множество экзофреймов, описывающих выявленные в тексте виды сообществ, их подвиды и т.д.

Таким образом, для выявления родовидовых отношений достаточно анализа графов семантических окрестностей термина.

Отношение «*Объект действия-действие-субъект действия*». Для распознавания в тексте данного отношения будем использовать множество модифицированных графов зависимостей $G'=\{g'\}$, построенных при пофразном синтаксическом анализе текста. Как при извлечении знаний из терминологических словарей, так и при извлечении знаний из научных текстов процесс распознавания отношения «Объект действия ↔ действие ↔ субъект действия» основан на анализе подграфов синтаксических конструкций графа зависимостей. Построение сети экзофреймов, описывающих отношение «Объект действия ↔ действие ↔ субъект действия», происходит аналогичным образом, как это было описано в пункте 2.5.2.2.

Таким образом, в результате анализа научного текста будет построена семантическая сеть экзофреймов, описывающих концептуальные объекты рассматриваемой предметной области. При этом некоторые экзофреймы останутся пустыми, то есть будут заполнены только имена фрейма и термина. Это означает, что данный термин является общенаучным для данной предметной области или относится к смежной предметной области.

2.5.3 Соединение онтологий

Построение онтологии предметной области выполняется на основе единого категориального аппарата, при этом вначале создается терминосистема, а затем – номенклатура [59]. Это означает, что при создании номенклатуры должна использоваться терминосистема, в которой определены все основные термины предметной области. Присоединение номенклатуры к терминосистеме может осуществляться как в процессе, так и после её создания.

При создании терминосистемы и номенклатуры заполняется их заголовок, который включает имя предметной области для терминосистемы и

имя области знаний для номенклатуры. Присоединение вновь создаваемой номенклатуры *NS* к существующей терминосистеме *TS* той же предметной области выполняется в процессе представления нового термина.

По каждому новому термину $t_k^{NS} \in Term^{NS}$, где $Term^{NS}$ – множество терминов номенклатуры *NS*, должен выполняться поиск по образцу данного термина в терминосистеме *TS*. Пусть имеем два образца τ_i^{TS} и τ_k^{NS}, соответствующие имени термина в терминосистеме *TS* и номенклатуре *NS*:

$$\tau_i^{TS} = \begin{pmatrix} z_1 & z_2 & z_3 \\ x_{i1} & x_{i2} & x_{i3} \end{pmatrix}, \ \tau_k^{NS} = \begin{pmatrix} z_1 & z_2 & z_3 \\ y_{k1} & y_{k2} & y_{k3} \end{pmatrix}, \tag{2.19}$$

где z_1 – имя термина;

z_2 – тип термина;

z_3 – вид сущности;

x_{ij} – значение t_i^{TS} для терминосистемы;

y_{kj} – значения для номенклатуры.

Если соответствующие x_{ij} и y_{kj} равны, то термину t_k^{NS} соответствует термин $t_i^{TS} \in Term^{TS}$, где $Term^{TS}$ – множество терминов терминосистемы *TS*.

Однако возможны случаи, когда в образце τ_k^{NS} неизвестны значения y_{k2} и/или y_{k3}, так как в научном тексте информация о термине может быть неполной. Тогда, если $x_{i1} = y_{k1}$, будем считать, что термину t_k^{NS} соответствует термин $t_i^{TS} \in Term^{TS}$. Если термин найден, то в соответствующую вершину семантической сети *G* или слота знака-фрейма *F* необходимо записать ссылку на термин t_i^{TS} терминосистемы.

Кроме того, необходимо добавить записи в заголовки терминосистемы и номенклатуры. В общем виде заголовки терминосистемы и номенклатуры имеют одинаковую структуру следующего вида:

```
<SubjectArea name = "…">
      <AttachedSA name="…">
         <Level value= "верхний"| "нижний"/>
         <Type value= "терминосистема"| "область знаний"|
                              "задача"| "вид деятельности"/>
         <LinkTerm name = "…"/>
      </AttachedSA>
…
</SubjectArea>.
```

Вполне возможно, что некоторые термины номенклатуры имеют имя, не совпадающее с именем термина в терминосистеме, но, по сути, являются квазинонимами. Поэтому после создания номенклатуры необходимо выполнить сравнение интенсионалов терминов номенклатуры и терминосистемы.

Интенсионал термина типа «Понятие» определяется кортежем:

$$T = <t_i, D, Pr, A, C, T_j, M>, \tag{2.20}$$

где t_i – имя термина, заданное вектором $Z = < z_{i1}, z_{i2}, z_{i3} >$;

D – множество дефиниций термина;

Pr – множество свойств термина;

A – множество действий;

C – кортеж, состоящий из множества синонимов и множества коррелятов термина;

T_j – множество терминов, находящихся в квалитативных отношениях с термином t_i;

M – множество метазнаков.

Так как элементами вектора T являются в основном множества, то анализ будем проводить по каждому элементу T отдельно, причем существенными будем считать множества Pr, T_j. Нельзя сказать, что множества C, M и A не существенны. Но, как правило, в научном тексте рассматриваются отдельные стороны термина, касающиеся какой-либо проблемы или задачи, поэтому будем считать, что для номенклатуры эти множества не существенны.

Для анализа интенсионалов будем использовать отношение сравнения элементов вектора T, которое рассмотрим по каждой паре терминов (t_i, t_j), такой, что $t_i^{TS} \in Term^{TS}$, $t_j^{NS} \in Term^{NS}$. Обозначим символом X множества вектора $Term^{TS}$, а Y – множества вектора $Term^{NS}$, то есть при рассмотрении элемента «Свойства» вектора T $X = Pr_i^{TS}$, а $Y = Pr_j^{NS}$, где $Pr_i^{TS} = \{pr_{i1}^{TS}, \ldots, pr_{in}^{TS}\}$ задает свойства термина $t_i^T \in Term^{TS}$, а множество $Pr_j^{NS} = \{pr_{j1}^{NS}, \ldots, pr_{jn}^{NS}\}$ задает свойства термина $t_i^{NS} \in Term^{NS}$.

При анализе множеств T_j^{TS} и T_j^{NS} отношение сравнения должно применяться для множеств терминов, связанных родовидовым отношением и отношением «часть-целое». При этом отдельно должны сравниваться множества родовых терминов, видовых терминов, терминов-целое, терминов-часть. Таким образом, в сравнении будут участвовать:

- множество свойств;
- множество родовых терминов;
- множество видовых терминов;
- множество терминов, означающих «целое»;
- множество терминов, означающих «часть».

Отношение сравнения множеств. Сравнение множеств X и Y будем осуществлять следующими отношениями: $Y \neq X$, $Y = X$, $Y \subset X$, $Y \supset X$, $Y \cap X \neq \varnothing$.

Если для любых X и Y отношение неравенства существует, то из этого следует, что термины t_i^{TS} и t_j^{NS} разные, и дополнительных действий не требуется, то есть номенклатура остается в той же конфигурации.

Если для любых X и Y отношение равенства существует, то это означает, что термины идентичны. В этом случае в знаке-фрейме F^{NT}, соответствующем данному термину в номенклатуре NT, необходимо удалить всю информацию кроме заголовочной. В заголовочную часть нужно добавить ссылку на знак-фрейм F^{ST}. Тогда в знаке-фрейме F^{NT} остается только имя термина t_j^{NS} и ссылка на t_i^{TS}. В том случае, если имена терминов не совпали, а для остальных множеств существует отношение равенства, то из этого следует, что термин t_j^{NS} является синонимом t_i^{TS}, и во множество синонимов терминосистемы нужно включить имя термина и ссылку на него.

Если для любых X и Y отношение включения $Y \subset X$ истинно, то это означает, что рассматриваемый термин t_j^{NS} наследует все свойства термина t_i^{TS}. Из этого следует, что в знаке-фрейме F^{NT}, соответствующем данному термину в номенклатуре NT, необходимо удалить всю информацию кроме заголовочной. В заголовочную часть добавляется ссылка на знак-фрейм F^{ST}.

Если для любых X и Y отношение включения $Y \supset X$ истинно, то это означает, что рассматриваемый термин t_j^{NS} обладает более полным описанием, чем термин t_i^{TS}, и его знак-фрейм F^{ST} необходимо дополнить недостающей информацией из знака-фрейма F^{NT}, затем удалить всю информацию из знака-фрейма F^{NT} кроме заголовочной. В заголовочную часть добавляется ссылка на знак-фрейм F^{ST}.

Надо отметить, что вероятность существования отношений равенства и включения на множествах X и Y невелика. Наиболее частым является случай, когда истинно отношение $Y \cap X \neq \varnothing$. Для его анализа лучше всего использовать аппарат нечеткой логики, который позволяет рассматривать различные ситуации. Например, возможны такие ситуации:

– часть свойств совпала в основном, и мощность конечного множества пересечения родовых понятий большая, мощность конечного множества пересечения видовых понятий небольшая, а мощности множеств пересечения других множеств ничтожно малы, то можно сказать, что термин t_i^{TS} в научном тексте рассматривается под другим углом. Этой ситуации, скорее всего, соответствует вывод, что в номенклатуре определен новый вид t_j^{NS} как отображение термина t_i^{TS};

– часть свойств совпала в основном, и мощность конечного множества пересечения родовых понятий небольшая, мощность конечного множества пересечения видовых понятий большая, а мощности множеств пересечения других множеств ничтожно малы, то можно сказать, что термин t_i^{TS} в научном тексте является «родом» понятия t_i^{TS};

– часть свойств совпала в основном, и мощности множеств пересечения других множеств ничтожно малы, то, скорее всего, термин t_j^{NS} является компонентом термина t_i^{TS}, то есть термином, означающим «часть»;

– и другие ситуации.

Таким образом, проекция терминосистемы TS на плоскость рассматриваемого научного текста в виде номенклатуры NS позволяет уточнять термины терминосистемы, определять новые виды терминов, новые компоненты терминов, то есть уточнять компьютерную терминологию предметной области.

Для реализации анализа отношения $Y \cap X \neq \varnothing$ хорошо подходят методы нечеткого регулирования. Рассмотрим основные аспекты их применения.

Отношение $Y_l \cap X_l \neq \varnothing$ может быть истинно для $\forall(Y_l, \forall X_l,)$ где l=1÷5. Индекс l последовательно нумерует множества в следующем порядке: множества свойств, множества родовых терминов, множества видовых терминов, множества терминов, означающих «целое» и множества терминов, означающих «часть». На практике могут встречаться различные комбинации

истинности отношения $Y_l \cap X_l \neq \varnothing$. Двумя наиболее интересными являются случаи, когда отношение $Y_l \cap X_l \neq \varnothing$ истинно при $l \in \{1,2,3\}$ и $l \in \{1,4,5\}$. В работе рассмотрим только первый случай.

Для нечеткого логического вывода анализа отношения $Y \cap X \neq \varnothing$ будем использовать метод нечеткого регулирования Мамдани [4], описание которого приведено в разделе 4.2.4.4. Компоненты нечеткого вывода рассмотрим на примере определения степени достоверности того, что термин t_j^{NS} является новым видом термина t_{i-1}^{TS}.

Основой для проведения операции нечеткого логического вывода является база правил, содержащая нечеткие высказывания в форме "Если-то" и функции принадлежности для соответствующих лингвистических термов. База правил нечетких продукций для определения степени достоверности того, что термин t_j^{NS} является новым видом термина t_{i-1}^{TS}, состоит из следующих элементов.

Имя *продукции* (N): определено как номер нечеткой продукции $N = 1,\ldots,21$.

Сфера *применения* (Q): выявление различия между терминами t_j^{NS} и t_{i-1}^{TS}.

Условие применимости (P): $(\forall Y_l \; \forall X_l)$ $Y_l \cap X_l \neq \varnothing$ истинно, где $l \in \{1,2,3\}$.

Условие ядра (A): составное нечеткое высказывание вида «$IS_i = t'$ И $DTIS_i = t'$ И $DNIS_i = t'$», где индекс $i \in \{1,2,3\}$ последовательно нумерует множества в следующем порядке: множества свойств, множества родовых терминов, множества видовых терминов;

IS_i – обозначения входных лингвистических переменных *InterSection*, соответствующих мощностям множеств $Y_l \cap X_l$;

$DTIS_i$ – обозначения входных лингвистических переменных *DifferenceTSInterSection*, соответствующих мощностям множеств $X_l/(Y_l \cap X_l)$;

$DNIS_i$ – обозначения входных лингвистических переменных *DifferenceNSInterSection*, соответствующих мощностям множеств $Y_l/(Y_l \cap X_l)$;

терм $t' \in T_1 =$ {Низкая (*Little*), Средняя (*Mean*), Высокая (*Big*)}.

Заключение ядра (B): нечеткое высказывание вида «$H\_NKT_i = \Delta t''$», где

H_NKT_1 – обозначение выходной лингвистической переменной СТЕПЕНЬ РАЗЛИЧИЯ СВОЙСТВ ТЕРМИНА;

H_NKT_2 – обозначение выходной лингвистической переменной СТЕПЕНЬ РАЗЛИЧИЯ РОДОВЫХ ПОНЯТИЙ ТЕРМИНА;

H_NKT_3 – обозначение выходной лингвистической переменной СТЕПЕНЬ РАЗЛИЧИЯ ВИДОВЫХ ПОНЯТИЙ ТЕРМИНА;

терм $t'' \in T_2$ и $T_2 =$ {Низкая (Low), Средняя (Normal), Высокая (High)}, модификатор $\Delta \in M_1$ и $M_1 =$ {Ниже (Down), Выше (Up)}.

База правил называется полной, если выполняются следующие условия:

1) существует хотя бы одно правило для каждого лингвистического терма выходной переменной;

2) для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки.

Для обеспечения выполнения данных условий необходимо в условии ядра представить полный перебор входных переменных. Количество последовательностей m длины L из M чисел рассчитывается по формуле:

$$m = M^L. \quad (2.21)$$

Так как условие ядра нечетких множеств лингвистических переменных состоит из трех переменных, и каждое правило использует каждую из переменных в качестве предпосылки, то было построено $m=3^3=27$ последовательностей длины 3 из переменных IS_i, $DTIS_i$, $DNIS_i$.

Анализ построенных правил показал, что не все правила могут быть использованы для проведения операции нечеткого логического вывода. Это связано с тем, что некоторые наборы значений входных лингвистических переменных IS_i, $DTIS_i$, $DNIS_i$ не имеют смысла в разрезе решаемой задачи. Например, рассмотрим условие ядра вида:

(*IS* is *big*) and (*DTIS* is *mean*) and (*DNIS* is *big*)

Это условие не может быть истинно, так как если мощность пересечения множеств $IS=Y_l \cap X_l$ большая, то очевидно, что мощности разности множеств $DTIS=X_l/(Y_l \cap X_l)$ и $DNIS=Y_l/(Y_l \cap X_l)$ не могут иметь большое или среднее значение. По аналогичным соображениям из рассмотрения были исключены шесть правил с условиями ядра вида:

1) (*IS* is *big*) and (*DTIS* is *big*) and (*DNIS* is *little*);
2) (*IS* is *big*) and (*DTIS* is *mean*) and (*DNIS* is *mean*);
3) (*IS* is *big*) and (*DTIS* is *big*) and (*DNIS* is *mean*);
4) (*IS* is *big*) and (*DTIS* is *mean*) and (*DNIS* is *big*);
5) (*IS* is *big*) and (*DTIS* is *little*) and (*DNIS* is *big*);
6) (*IS* is) and (*DTIS* is *big*) and (*DNIS* is *big*).

Таким образом, база правил содержит 21 нечеткое правило R_i, то есть $i=1 \div m'$, $m'=21$, и является полной. На рисунке 2.9 показан фрагмент базы правил.

1. If (IS is little) and (DTIS is little) and (DNIS is little) then (H_NKT is low) (1)
2. If (IS is mean) and (DTIS is little) and (DNIS is little) then (H_NKT is low) (1)
3. If (IS is big) and (DTIS is little) and (DNIS is little) then (H_NKT is low) (1)
4. If (IS is little) and (DTIS is mean) and (DNIS is little) then (H_NKT is low) (1)
5. If (IS is mean) and (DTIS is mean) and (DNIS is little) then (H_NKT is low) (1)
6. If (IS is big) and (DTIS is mean) and (DNIS is little) then (H_NKT is low) (1)
7. If (IS is little) and (DTIS is big) and (DNIS is little) then (H_NKT is up_low) (1)
8. If (IS is mean) and (DTIS is big) and (DNIS is little) then (H_NKT is up_low) (1)
9. If (IS is little) and (DTIS is little) and (DNIS is mean) then (H_NKT is down_norm) (1)
10. If (IS is mean) and (DTIS is little) and (DNIS is mean) then (H_NKT is down_norm) (1)
11. If (IS is big) and (DTIS is little) and (DNIS is mean) then (H_NKT is up_low) (1)
12. If (IS is little) and (DTIS is mean) and (DNIS is mean) then (H_NKT is norm) (1)
13. If (IS is mean) and (DTIS is mean) and (DNIS is mean) then (H_NKT is norm) (1)
14. If (IS is little) and (DTIS is big) and (DNIS is mean) then (H_NKT is up_norm) (1)
15. If (IS is mean) and (DTIS is big) and (DNIS is mean) then (H_NKT is norm) (1)
16. If (IS is little) and (DTIS is little) and (DNIS is big) then (H_NKT is up_norm) (1)
17. If (IS is mean) and (DTIS is little) and (DNIS is big) then (H_NKT is up_norm) (1)
18. If (IS is little) and (DTIS is mean) and (DNIS is big) then (H_NKT is down_high) (1)

Рисунок 2.9 – Фрагмент базы правил определения степени различия свойств терминов

Используемый метод нечеткого регулирования Мамдани основан на минимаксной композиции нечетких множеств и включает в себя следующую последовательность действий.

На этапе фазификации определяются степени истинности условий ядра каждого правила. Для базы правил с m' правилами обозначим степени истинности как $\mu_i(\varphi)$, где $i=1\div m'$, φ – входные переменные.

Количественное значение степени истинности заключения ядра (S) определяется посредством метода min-активизации:

$$\mu_i'(y) = \min\{\mu_i(\varphi), \mu_i(y)\}, \qquad (2.22)$$

где $\mu(y)$ – функция принадлежности терма, который является значением выходных лингвистических переменных H_NKT_i, заданных на универсуме $Y = [0;1]$, $i = \overline{1,21}$.

Процедура агрегирования нечеткого логического вывода сводится к выбору максимального значения функций истинности:

$$MF(y) = \max_i\{\mu_i(y)\}, \qquad (2.23)$$

где $MF(y)$ – функция принадлежности итогового нечеткого множества, $i = \overline{1,21}$.

Дефаззификация проводится методом среднего центра, или центроидным методом (*centroid*). В результате выходные переменные H_NKT_i получают четкие значения, определяющие степень различия множеств свойств, родовых понятий, видовых понятий терминов t_j^{NS} и t_{i-1}^{TS}.

Чтобы определить вид зависимостей функции принадлежности нечетких множеств, нами рассмотрены различные виды функций принадлежности. В таблице 2.6 приведена выборка результатов нечеткого логического вывода по Мамдани с треугольной (*trimf*), гауссовой (*gaussmf*) и колоколообразной (*gbellmf*) функциями принадлежностей на примере сравнения множеств свойств термина. Здесь N – номер набора значений входных переменных IS_1, $DTIS_1$, $DNIS_1$.

В результате анализа была выбрана колоколообразная функция принадлежности.

Посредством построенной базы правил рассчитываются СТЕПЕНЬ РАЗЛИЧИЯ РОДОВЫХ ПОНЯТИЙ ТЕРМИНА H_NKT_2 и СТЕПЕНЬ РАЗЛИЧИЯ ВИДОВЫХ ПОНЯТИЙ ТЕРМИНА H_NKT_3.

Для определения СТЕПЕНИ ДОСТОВЕРНОСТИ ГИПОТЕЗЫ О НОВОМ ВИДЕ ТЕРМИНА используется функция выбора максимального значения функций принадлежностей, рассчитанных для множества свойств, множества родовых понятий, множества понятий терминов:

$$Out_i = \max_i\{H\_NKT_{1i}, H\_NKT_{2i}, H\_NKT_{3i}\}. \qquad (2.24)$$

Рассмотрим определение степени достоверности того, что термин t_j^{NS} является новым видом термина t_i^{TS} на примере сравнения словарных статей терминосистемы и номенклатуры термина t=«ЗАРАБОТНАЯ ПЛАТА». Словарная

статья терминосистемы построена на основе извлечения знаний из экономического терминологического словаря [8], а номенклатуры – из лекции «Определение среднего уровня заработной платы» курса «Экономическая теория». В приложении В приведены словарные статьи в формате XML.

Таблица 2.6 – Результаты нечеткого логического вывода для множества свойств

N	IS_1	$DTIS_1$	$DNIS_1$	H_NKT_1		
				trimf	**gaussmf**	**gbellmf**
1	0,50	0,00	0,50	**0,412**	**0,417**	**0,477**
2	0,50	0,50	0,00	**0,133**	**0,152**	**0,117**
3	1,00	0,00	0,00	**0,133**	**0,153**	**0,087**
4	0,50	0,25	0,25	**0,411**	**0,420**	**0,427**
5	0,75	0,25	0,00	**0,167**	**0,207**	**0,138**
6	0,75	0,00	0,25	**0,405**	**0,413**	**0,427**
7	0,63	0,13	0,25	**0,405**	**0,414**	**0,427**
8	0,63	0,25	0,13	**0,274**	**0,321**	**0,228**
9	0,63	0,38	0,00	**0,145**	**0,164**	**0,088**
10	0,67	0,11	0,22	**0,375**	**0,375**	**0,324**
11	0,67	0,00	0,33	**0,414**	**0,420**	**0,483**
12	0,67	0,33	0,00	**0,152**	**0,175**	**0,095**
13	0,70	0,10	0,20	**0,350**	**0,346**	**0,249**
14	0,70	0,20	0,10	**0,157**	**0,253**	**0,135**
15	0,70	0,00	0,30	**0,413**	**0,421**	**0,479**
16	0,70	0,30	0,00	**0,157**	**0,185**	**0,106**
17	0,80	0,10	0,10	**0,157**	**0,253**	**0,135**
18	0,80	0,00	0,20	**0,350**	**0,346**	**0,249**
19	0,80	0,20	0,00	**0,157**	**0,185**	**0,106**
20	0,90	0,00	0,10	**0,141**	**0,213**	**0,107**
21	0,90	0,10	0,00	**0,141**	**0,158**	**0,087**
22	1,00	0,00	0,00	**0,133**	**0,152**	**0,087**

Введем следующие обозначения:

1) $X_1=Pr^{TS}$, а $Y_1=Pr^{NS}$, где $Pr^{TS} = \{pr_i^{TS}\}$ задает свойства термина $t \in Term^{TS}$, $Pr^{NS}= \{pr_j^{NS}\}$ задает свойства термина $t \in Term^{NS}$;

2) $X_2=T_1^{TS}$, а $Y_2=T_1^{NS}$, где $T_1^{TS} = \{t_{1i}^{TS}\}$ задает родовые понятия термина $t \in Term^{TS}$, $T_1^{NS}= \{t_{1j}^{NS}\}$ задает родовые понятия термина $t \in Term^{NS}$;

3) $X_3=T_2^{TS}$, а $Y_3=T_2^{NS}$, где $T_2^{TS} = \{t_{2i}^{TS}\}$ задает видовые понятия термина $t \in Term^{TS}$, $T_2^{NS}= \{t_{2j}^{NS}\}$ задает видовые понятия термина $t \in Term^{NS}$.

Таким образом, для рассматриваемого термина t=«ЗАРАБОТНАЯ ПЛАТА»:

X_1={размер минимальной заработной платы, единая тарифная сетка};

X_2={денежное вознаграждение};

X_3={сдельная, повременная, аккордная};

Y_1={размер минимальной заработной платы, уровень заработной платы, рост заработной платы, ставка заработной платы, средняя заработная плата};

Y_2={денежное вознаграждение};

Y_3={сдельная, повременная, аккордная, номинальная, реальная}.

Тогда для множества свойств термина t: $X_1 \cap Y_1$ ={размер минимальной заработной платы}; $X_1/X_1 \cap Y_1$={единая тарифная сетка}; $Y_1/X_1 \cap Y_1$={уровень заработной платы, рост заработной платы, ставка заработной платы, средняя заработная плата}.

Для множества родовых понятий термина t: $X_2 \cap Y_2$ ={денежное вознаграждение}; $X_2/X_2 \cap Y_2=\varnothing$; $Y_2/X_2 \cap Y_2=\varnothing$.

Для множества видовых понятий термина t: $X_3 \cap Y_3$ ={сдельная, повременная, аккордная}; $X_3/X_3 \cap Y_3=\varnothing$; $Y_3/X_3 \cap Y_3$={номинальная, реальная}.

То есть мощности данных множеств соответственно равны:

$|X_1 \cap Y_1|=1$, $|X_1/X_1 \cap Y_1|=1$, $|Y_1/X_1 \cap Y_1|=4$;

$|X_2 \cap Y_2|=1$, $|X_2/X_2 \cap Y_2|=0$, $|Y_2/X_2 \cap Y_2|=0$;

$|X_3 \cap Y_3|=3$, $|X_3/X_3 \cap Y_3|=0$, $|Y_3/X_3 \cap Y_3|=2$.

Определим значения входных лингвистических переменных. Для этого найдем мощности объединений соответствующих множеств:

$|X_1 \cup Y_1|=6$, $|X_2 \cup Y_2|=1$, $|X_3 \cup Y_3|=5$.

Далее рассчитаем значения входных лингвистических переменных как долю от общего количества свойств, родовых понятий, видовых понятий соответственно:

$IS_i = |X_i \cap Y_i|/|X_i \cup Y_i|$,

$DTIS_i = |X_i/X_i \cap Y_i|/|X_i \cup Y_i|$,

$DNIS_i = |Y_i/X_i \cap Y_i|/|X_i \cup Y_i|$,

где $i \in \{1,2,3\}$.

Тогда получаем:

IS_1=0.17, $DTIS_1$=0.17, $DNIS_1$=0,67;

IS_2=1, $DTIS_2$=0, $DNIS_2$=0;

IS_3=0.6, $DTIS_3$=0, $DNIS_3$=0.33.

Используя разработанную базу правил, реализованную в среде Matlab, найдем выходные переменные: H_NKT_1 = 0.487; H_NKT_2 = 0.087; H_NKT_3 = 0.483.

На рисунке 2.10 изображены нечеткие множества (закрашенная область) выходных лингвистических переменных *Степень различия свойств термина* H_NKT_1 (рис. 2.10а), *Степень различия родовых понятий* H_NKT_2 (рис. 2.10б), *Степень различия видовых понятий термина* H_NKT_3 (рис. 2.10в).

а б в

Рисунок 2.10 – График функции принадлежности терма выходной лингвистической переменной H_NKT_i

Согласно формуле 2.20 СТЕПЕНЬ ДОСТОВЕРНОСТИ ГИПОТЕЗЫ О НОВОМ ВИДЕ ТЕРМИНА равна: Out=max{0.487, 0.087, 0.483}=0.487.

Таким образом, СТЕПЕНЬ ДОСТОВЕРНОСТИ ГИПОТЕЗЫ О НОВОМ ВИДЕ ТЕРМИНА средняя, то есть результат нечеткого логического вывода можно интерпретировать следующим образом: термин *t*=«ЗАРАБОТНАЯ ПЛАТА» в научном тексте рассматривается под другим углом в сравнении с термином *t*=«ЗАРАБОТНАЯ ПЛАТА» в терминологическом словаре. Действительно, в словаре дано общее определение термина, а в статье заработная плата рассматривается с точки зрения спроса и предложения труда в условиях совершенной конкуренции.

2.6 Выводы по главе

Автоматическое извлечение знаний из научных текстов предполагает не только выявление терминов, но и извлечение знаний о них. Для этого необходимо распознать в тексте семантические отношения между терминами, так как они задают семантическую структуру терминологии. В связи с этим в главе построена иерархия семантических отношений посредством применения классификации концептуальных объектов. Главным основанием классификации семантических отношений, предложенной в работе, является их принадлежность понятийным сферам. Такой подход позволил значительно сократить количество типов предикатов, что достаточно важно для системы резолютивного логического вывода. Приведено соответствие семантических отношений и типов предикатов. Предикаты являются ядром продукционного правила. Приведено обоснование модели представления знаний о методе естественно-языковой обработки информации в виде систем продукций на основе аппарата ситуационного управления.

Большой материал посвящен методам естественно-языкового анализа научного текста, которые представлены в виде систем продукций. Эти методы позволяют построить семантическую сеть знаков-фреймов терминосистемы и номенклатуры. Рассмотрены основные аспекты соединения онтологических моделей терминосистемы и номенклатуры предметной области. Предложен способ соединения, основанный на анализе отношений неравенства, равенства, включения и пересечения множеств, представляющих элементы интенсионала сравниваемых терминов.

Предложенный подход соединения онтологий позволяет объединять онтологии одной предметной области на основе анализа интенсионалов терминов, принадлежащих терминосистеме и номенклатуре. Для этого выполняется сравнение отношений между соответствующими множествами интенсионалов. Нечеткость мощности конечного множества пересечения обусловила использование нечеткого логического вывода. В работе показан способ введения нечеткости и применение методов нечеткого регулирования.

3. Генерация систем продукций как декларативных методов естественно-языковой обработки монологического текста

Для задач, обладающих слабоструктурированными предметными областями, обычно отсутствует объективная модель их решения. Как правило, такие задачи решаются экспертами. Автоматическое создание правил решения задачи на основе конструктивных знаний эксперта позволит сформировать библиотеку декларативных методов, обладающую свойствами долговечности и масштабируемости. Это актуально, так как экспертам зачастую трудно сформулировать правила, которыми они пользуются при решении задач, поскольку экспертное знание в большинстве случаев является подсознательным. Именно подсознательный характер экспертного знания вызывает трудности при построении экспертных систем, а извлечение экспертных знаний считается «узким местом» искусственного интеллекта [5,130]. В работе [42] для выявления экспертных решающих правил в базах знаний использована методология эволюционного моделирования. Предложенный в работе О.И Ларичева генетический алгоритм позволяет восстановить решающие правила, которые подсознательно или осознанно использовались экспертом при решении задач классификации.

В данной работе предлагается генетический алгоритм для генерации ядер продукционных правил, предназначенных для естественно-языковой обработки научного текста.

3.1 Конструктивные знания эксперта и формирование спецификации предметной области прикладной задачи

В соответствии с выводами раздела 2.1 предлагается автоматическая генерация системы продукций на основе конструктивных знаний эксперта.

3.1.1 Конструктивные знания

Конструктивные знания – это знания о наборах возможных структур объектов и взаимодействии между их частями. Модель предметной области M_{SA} можно определить как совокупность понятийных и конструктивных знаний в виде кортежа [75]:

$$M_{\mathrm{SA}} = <T_k, K_k>, \qquad (3.1)$$

где T_k – множество понятий;

K_k – кортеж, описывающий множество конструктов и их взаимосвязи.

3.1.1.1 Понятия и конструкты

Компонентами кортежа K_k являются множество конструктов K_{kc}, семейство множеств допустимых значений элементов конструктов K_{kv}, множество графов K_{kg}, определяющих допустимые взаимосвязи конструктов.

В принципе при анализе естественно-языкового текста конструктивные знания тесно связаны как с самими языковыми конструкциями, так и с

языковыми конструкциями более высокого порядка, предназначенными для распознавания первых. Рассмотрим создание модели M_{SA} на примере метода извлечения знаний из терминологических словарей о квалитативном отношении «Целое-часть».

Определим прежде всего понятийное множество T_k метода, которое включает следующие понятия: предложение p, композиционное словосочетание k (КСС), именное субстантивное словосочетание s (ИСС), семантическое отношение r (СемОтношение), заголовочный термин z, термин t, лексема l, список q, элемент списка e (ЭлСписка), глагол v, терм-спутникR tr, терм-спутникX tx, терм-спутникY ty, признак h, характеристика x, часть речи c_1 (ЧастьРечи), падеж c_2, значение.

В данном методе объектом исследования является предложение, которое содержит некоторые компоненты, которые, в свою очередь, могут включать в себя другие компоненты. Все компоненты могут иметь характеристики, которые соответствуют конкретным свойствам, имеющим некоторые значения. Тогда в зависимости от ситуации, которая определяется значениями характеристик и взаимным расположением компонентов, можно определить, задано ли в предложении между некоторыми компонентами семантическое отношение «Целое-часть».

Сформируем множество конструктов K_{kc} как множество отношений R_i:

$$K_{kc}=\{R_i|\ R_i=\{(x,y)|\ x\in X;\ y\in Y;\ X\cap Y\neq\varnothing;\ i=1\div n\}, \qquad (3.2)$$

где X, Y – множества понятий, используемых в методе.

Для рассматриваемого метода n=10. Тогда в общем виде множество K_{kc} будут составлять следующие отношения:

$R_1=\{(x,y)|\ x$ содержит y, $x\in\{$Предложение$\}$, $y\in A\}$;

$R_2=\{(x,y)|\ x$ содержит y, $x\in B$, $y\in C\}$;

$R_3=\{(x,y)|\ x$ содержит y, $x\in\{$ИСС$\}$, $y\in D\}$;

$R_4=\{(x,y)|\ x$ содержит y, $x\in\{$СемОтношение$\}$, $y\in E\}$;

$R_5=\{(x,y)|\ x$ имеет y, $x\in F$, $y\in\{$Характеристика$\}\}$;

$R_6=\{(x,y)|\ x$ есть y, $x\in\{$Характеристика$\}$, $y\in G\}$;

$R_7=\{(x,y)|\ x$ есть y, $x\in\{$ЭлементСписка$\}$, $y\in H\}$;

$R_8=\{(x,y)|\ x$ имеет y, $x\in I$, $y\in\{$Значение$\}\}$;

$R_9=\{(x,y)|\ x$ имеет y, $x\in A$, $y\in\{$Индекс$\}\}$.

$R_{10}=\{(x,y)|\ x$ эквивалентен x, $x\in J\}\}$.

Каждый метод будет иметь собственное семейство множеств конструктов K_{kc}, которое должно быть конкретизировано. Для конкретизации представим отношение в виде xRy.

Тогда при A= {КСС, ИСС, СемОтношение, термин, лексема, список} первое множество будет иметь вид:

$R_1=\{$<Предложение> содержит <КСС>,
<Предложение> содержит <ИСС>,
<Предложение> содержит <СемОтношение>,
<Предложение> содержит <Термин>,
<Предложение> содержит <Лексему>,

<Предложение> содержит <Признак>}.

Второй конструкт при B= {КСС, список} и *C*= {ИСС, ЭлементСписка} будет следующим:

R_2={<КСС> содержит <ИСС>,
<Список> содержит <ЭлементСписка>}.

Третий конструкт при *D* = {Термин, Лексема, Терм-спутникX, Терм-спутникY}:

R_3={<ИССловоСоч> содержит <Термин>,
<ИССловоСоч> содержит < Лексема >,
<ИССловоСоч> содержит < Терм-спутникX >,
<ИССловоСоч> содержит < Терм-спутникY>}.

Четвертый конструкт при *E*= {Глагол, Терм-спутникR}:

R_4={<СемОтношение> содержит <Глагол>,
<СемОтношение> содержит <Терм-спутникR>}.

Первые четыре множества конструктов отражают возможную иерархическую структуру понятий предметной области. Пятый конструкт позволяет задать характеристики понятий, если они имеются. Для данного метода важны характеристики следующих понятий *H*={ИСС, Термин, Лексема, ЭлементСписка, Терм-спутникX, Терм-спутникY }:

R_5 = {<ИСС> имеет <Характеристика>,
<Термин> имеет <Характеристика>,
<Лексема > имеет <Характеристика>,
<ЭлементСписка> имеет <Характеристика>,
<Терм-спутникX> имеет <Характеристика>,
<Терм-спутникY> имеет <Характеристика>}.

Следующий конструкт уточняет значение характеристики как категории. При *G* = {Падеж и ЧастьРечи} имеем следующее множество конструктов:

R_6 = {<Характеристика> есть < Падеж>,
<Характеристика> есть <ЧастьРечи>}.

Седьмой конструкт также уточняет значение понятия «Элемент списка» как категории. При *H*={ИСС, Термин, Лексема} имеем:

R_7 = {<ЭлементСписка> есть <ИСС>,
<ЭлементСписка> есть <Термин>,
<ЭлементСписка> есть <Лексема>}.

Восьмой конструкт необходим для задания конкретных значений характеристикам понятий и некоторым вспомогательным понятиям, на основании значений которых и делается некоторый вывод. При *I* = {Глагол, Терм-спутникR, Терм-спутникX Терм-спутникY, Признак, ЧастьРечи, Падеж} имеем:

R_8 = {<Глагол> имеет <Значение>,
<Терм-спутникR> имеет <Значение>,
<Терм-спутникX> имеет <Значение>,
<Терм-спутникY> имеет <Значение>,
<Признак > имеет <Значение>,

<Падеж> имеет <Значение>,
<ЧастьРечи> имеет <Значение>}.

Для определения местоположения компонентов в предложении используется девятый конструкт, задающий положение основных понятий в предложении посредством индекса:

R_9 = {<КСС> имеет <Индекс>,
<ИСС> имеет <Индекс>,
<СемОтношение> имеет <Индекс>,
<Термин> имеет <Индекс>,
<Лексема> имеет <Индекс>,
<Список> имеет <Индекс>}.

В данном конструкте в качестве левой части используется множество *A*, задающее основные компоненты предложения. Обычно этого достаточно для распознавания семантических отношений между компонентами.

Последний конструкт связан с введением отношения тождества между компонентами, которое в данном методе необходимо для проверки значений характеристик или индексов (*J*={Падеж, ЧастьРечи, Индекс }):

R_{10} = {<Падеж> эквивалентен <Падеж>,
<ЧастьРечи> эквивалентен <ЧастьРечи>,
<Индекс> эквивалентен <Индекс>}.

Например, имеются два компонента ИСС и СемОтношение, которые соответственно имеют индексы *i*, *j*. Если ИСС должен следовать за СемОтношение, то это можно проверить следующим отношением:

"*j* эквивалентен (*i*+1)".

Таким образом, создается некоторая иерархическая организация, которая хорошо согласуется с теоретическими основаниями когнитивной психологии, согласно которой при мышлении используются не языковые конструкции как таковые, а их коды в форме некоторых абстракций, которые образуют иерархические структуры. Далее можно определить возможное следование конструктов.

3.1.1.2 Взаимосвязь конструктов

Рассмотрим взаимосвязь конструктов на примере. Допустим, что предложение *p* содержит композиционное словосочетание k_1, семантическое отношение *r* и список некоторых элементов *q*. Композиционное словосочетание *k*, в свою очередь, содержит (состоит) два именных субстантивных словосочетания s_1 и s_2. Первое словосочетание s_1 имеет характеристику x_1, которая является (есть) падежом c_1 со значением «Именительный». Второе словосочетание s_2 содержит заголовочный термин z_1, который имеет характеристику x_2, которая является (есть) падежом c_2 со значением «Родительный».

Семантическое отношение *r* содержит глагол *v* и терм-спутникR *tr*. При этом *v* имеет значение «состоит», а *tr* – «из частей».

Список q имеет элемент списка e, который является именным субстантивным словосочетанием s_3, имеет характеристику x_3, которая является частью речи c_3 со значением «Существительное».

Кроме того, композиционное словосочетание k_1, семантическое отношение r и список q имеют индексы i, j, l. При этом индекс j эквивалентен индексу (i+1), а индекс l эквивалентен индексу (i+2).

Рассмотренный пример можно представить в виде следующей графовой структуры, изображенной на рисунке 3.1.

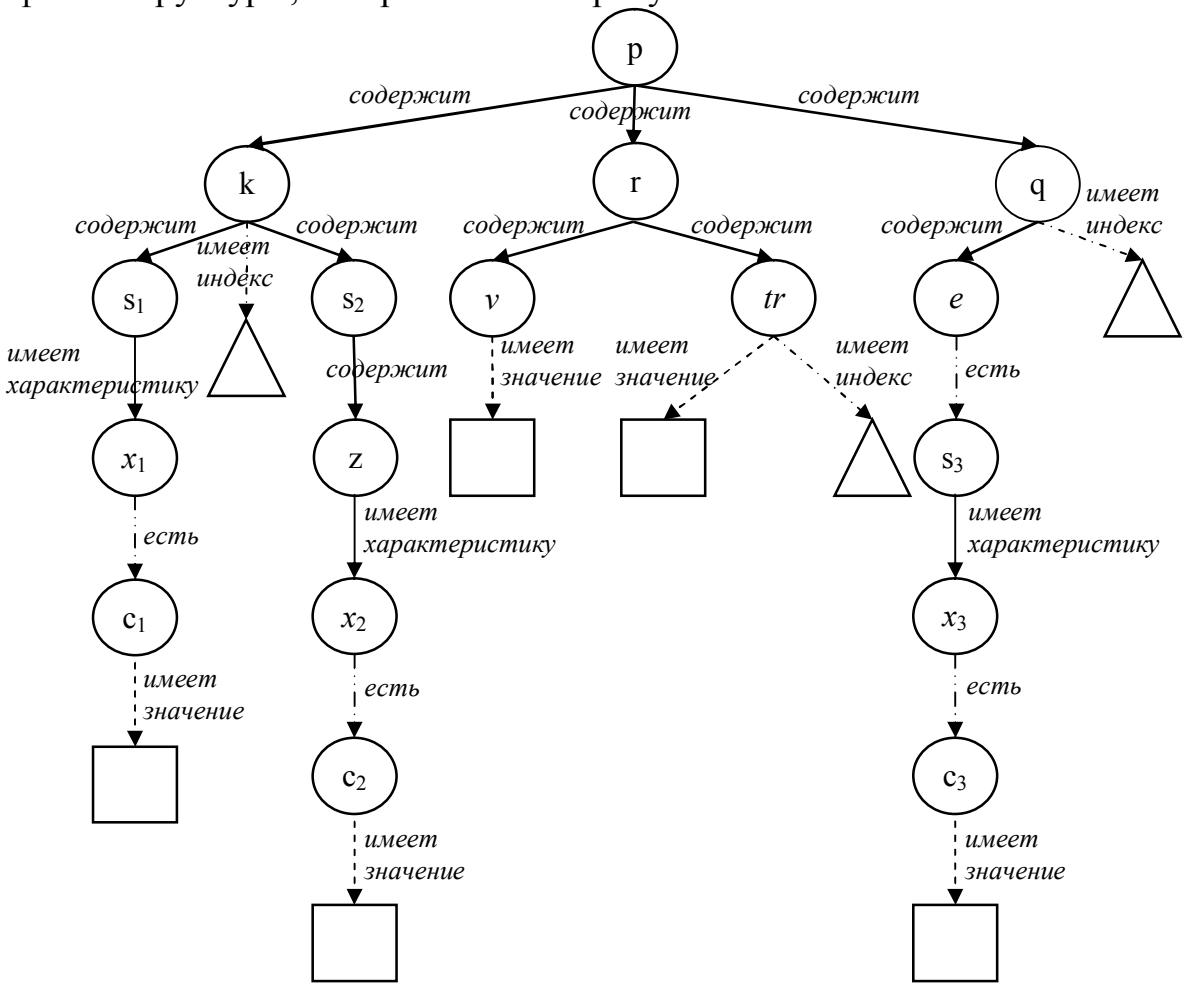


Рисунок 3.1 – Пример графовой структуры, отражающей взаимосвязь конструктов метода извлечения знаний о семантическом отношении «Целое-часть»

Таким образом, для рассматриваемого метода в корне графа всегда находится объект исследования (для данного метода это – предложение). Вершины первого уровня графа содержат основные компоненты предложения. Верхние дуги графа в основном помечены глаголом «содержит», что показывает иерархическую вложенность понятий. Листья графа содержат константные значения или индекс.

Из анализа существующих продукций (Приложение Б), разработанных для данного метода, было выявлено, что первый уровень вершин V_1 графа (дерева) могут составлять множество из одиннадцати альтернативных наборов компонентов:

$$V_1 = \{(k, r, q|s|l), (s, r, q|s|l), (z, r, q|s|t), (s|z, r, q, h)\}. \quad (3.3)$$

Это множество определяет возможные ветки дерева: k, s, z, r, q, l, t, h.

Возможные варианты построения веток метода извлечения знаний о семантическом отношении «Целое-часть» приведены ниже.

$$branch\_k = \begin{cases} s(tx(value)), \quad s(z(x(c(value)))) \\ s(x(c(value))), \quad s(z(x(c(value)))), index \\ s(tx(value, (x(c(value)))), \quad s(z(x(c(value)))), index \end{cases}$$

$$branch\_s = \begin{cases} index \\ z(x(c(value))), \quad index \\ l(x(c(value)), x(c(value))), \quad index \\ z(x(c(value))), \quad tx(x(c(value))), \quad index \\ l(x(c(value))), \quad ty(value), \quad index \end{cases}$$

$$branch\_z = \begin{cases} index \\ x(c(value), \quad index \end{cases}$$

$$branch\_r = \begin{cases} v(value) \\ v(value), \quad index \\ v(value), \quad tr(value) \\ v(value), \quad tr(value), \quad index \end{cases}$$

$$branch\_q = \begin{cases} e(x(c(value))), \quad index \\ e(x(c(value))), \quad e(x(c(value))), \quad index \\ e(s(x(c(value)))), \quad index \\ e(s(x(c(value)))), \quad e(x(c(value))), \quad index \end{cases}$$

$$branch\_l = \begin{cases} x(c(value)), \quad index \\ x(c(value)), \quad x(c(value)), \quad index \end{cases}$$

$$branch\_t = \begin{cases} x(c(value)), \quad index \\ x(c(value)), \quad x(c(value)), \quad index \end{cases}$$

$$branch\_h = \begin{cases} h(value) \\ h(value), \quad index \end{cases}$$

Для лучшего понимания приведенного описания рассмотрим в первой ветке *branch_k* второй вариант: $<s(x(c(value))), \quad s(z(x(c(value)))), index>$. Он соответствует изображению ветки *k* на рисунке 3.1. Запись варианта означает, что вершина *k* распадается на две ветки с вершинами *s* и ветку с

вершиной *index*, которая является листом. Ветки с вершинами *s* представляют собой последовательно расположенные вершины, заканчивающиеся листьями с константным значением.

Для обобщения результатов описания спецификации предметной области решаемой задачи (метода) построим конструкты ещё для одного метода.

3.1.1.3 Пример конструктов и их взаимосвязей

Рассмотрим метод морфологического анализа «Определение части речи». В этом методе объектом анализа является лексема. Лексема может быть готовой словоформой, в противном случае лексема разбивается на основу и окончание, анализ которых приводит к определению искомого результата.

В данном случае первый конструкт будет выглядеть следующим образом:

R_1= {(x,y)| x принадлежит y, $x \in A$, $y \in B$}, где

A = {Лексема, Основа, Окончание}, B = {Словарь Готовых словоформ (СГС), Словарь основ (СОС), Словарь Окончаний (СОК)}. В полном виде

R_1= {<Лексема> принадлежит <словарь СГС>,
<основа> принадлежит <словарьОснов>,
<окончание> принадлежит <словарьОкончаний>}.

Данный конструкт используется первым только при поиске лексемы в словаре готовых словоформ, в остальных случаях первым используется следующий конструкт:

R_2= {(x,y)| x содержит y, $x \in C$, $y \in D$}, где

C = {Лексема, СГС, СОС, СОК}, D={Основа, Окончание, Вектор, Компонент}. В полном виде

R_2= {<Лексема> содержит <основа>,
<Лексема> содержит <окончание>,
<СГС> содержит <вектор>,
<СОС> содержит <вектор>,
<СОК> содержит <вектор>}.

Следующие три конструкта направлены на идентификацию словаря:

R_3={<Словарь> имеет <Характеристика>};
R_4={<Характеристика> есть <имя>};
R_5={<имя> имеет <значение>}.

Далее следуют конструкты, которые описывают вектор:

R_6={<Вектор> содержит <Компонент>};
R_7={(x,y)| x есть y, x$\in${Компонент}, $y \in E$}, где
E={ГотСловоформа, ЧастьРечи, Основа, Окончание}, тогда
R_7={<Компонент> есть <ГотСловоформа>,
<Компонент> есть <ЧастьРечи>,
<Компонент> есть <Основа>,
<Компонент> есть <Окончание>}.

R_{10}={(x,y)| x эквивалентен y, x∈A, y∈F }, где
F={ГотСловоформа, Основа, Окончание }, тогда
R_{10}= {<Лексема> эквивалентен <ГотСловоформа>,
<Основа> эквивалентен <Основа>,
<Окончание> эквивалентен <Окончание>}.

Во всех методах морфологического анализа корнем дерева является лексема l как объект исследования, и в основном все методы морфологического анализа начинаются со второго конструкта R_2, определяющего состав компонентов. Поэтому на первом уровне имеются, как правило, две вершины «основа» b и «окончание» o. Исключение составляет только метод поиска лексемы в словаре готовых словоформ (рис. 3.2).

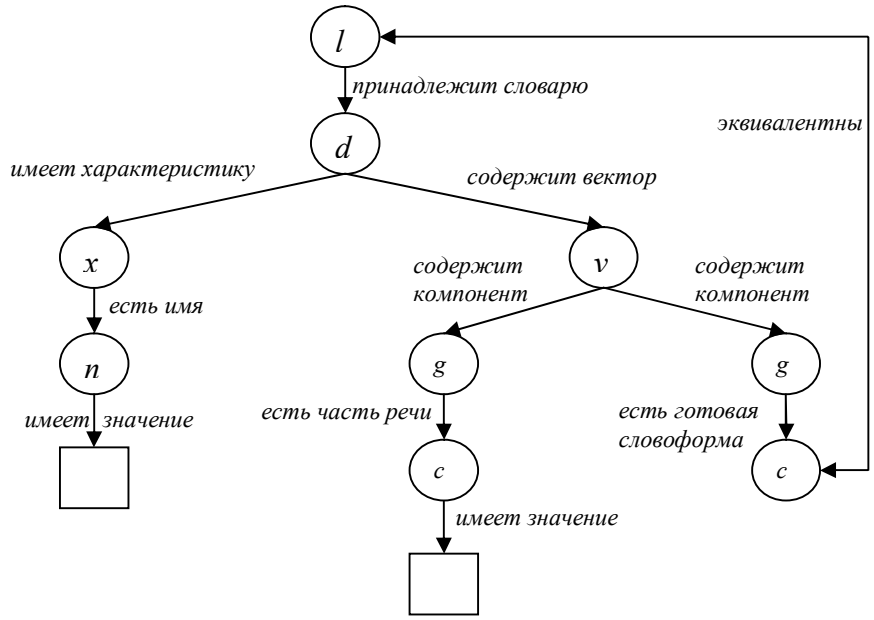


Рисунок 3.2 – Пример графовой структуры, отражающей взаимосвязь конструктов метода морфологического анализа «Поиск готовой словоформы»

Опишем допустимое множество деревьев для метода определения части речи. Приведем множество альтернативных кортежей, компоненты которых составляют вершины первого уровня

V_1= {d, (b, o), b, (d, x, x), (o, x, x)}.

Это множество определяет возможные ветки дерева: d, b, o, x.

$$branch\_d = \begin{cases} x(n(value)),\ v\_RWord(g_1(c(value))),\ g_2(c(l))) \\ x(n(value)),\ v\_SMI(g_1(c_1(c_5)),\ g_2(c_2(c_6)),\ g_3(c_3),\ g_4(c_4)) \end{cases}$$

$$branch\_b = \begin{cases} d(x(n(value)),\ v\_Base(g_1(c_1(value, c_3)),\ g_2(c_2(b)))) \\ d(x(n(value)),\ v\_FC(g_1(c_1),\ g_2(c_2(b)))) \end{cases}$$

$$branch\_o = \begin{cases} d(x(n(value)),\ v\_End(g_3(c_3(value)),\ g_4(c_4(o)))) \\ d(x(n(value)),\ v\_DMI(g_1(c_1(c_7)), g_2(c_2(c_8)), g_3(c_3(o)), g_4(c_4), g_5(c_5), g_6(c_6))) \end{cases}$$

$$branch\_x = \begin{cases} x(c) \\ x(c(value)) \end{cases}.$$

Таким образом, каждое продукционное правило может быть представлено в виде дерева. Причем дерево строится на основе конструктов, каждый из которых состоит из двух вершин и помеченной дуги. Две вершины, содержащие конкретные понятия, связаны дугой с точно заданной пометой (семантическое отношение). Допустимое соединение конструктов задается с помощью описаний веток дерева.

Из этого следует, что спецификация метода включает:

– множество понятий T_k, используемых в методе;

– семейство множеств конструктов K_{kc}, определяющих связь между двумя понятиями;

– множество графов K_{kg} в виде множества альтернативных кортежей, компоненты которых составляют вершины первого уровня графа и множества возможных вариантов построения веток.

Этой информации достаточно для генерации ядер продукционных правил, входящих в систему продукций, являющейся декларативным представлением метода. Рассмотрим средство описания спецификации.

3.1.2 XML-описание спецификации метода

Язык XML призван стать *lingua franca* **(**универсальным языком) для обмена информацией среди пользователей и программ [12,29]. В соответствии с этой концепцией пользователи, а также специализированные программы должны иметь возможность создавать и прочитывать XML-документы. Доступность и прозрачность языка для пользователя выделяют XML из большинства других форматов, применяемых при построении текстовых документов и баз данных.

Основное достоинство языков разметки и, в частности, языка XML заключается в том, что в описании объединены данные и метаданные о данных. Причем метаданные обеспечивают необходимую структурированность данных. К тому же хранение данных вместе с метаданными позволяет в большей степени абстрагироваться от конкретного метода. Недостатком данного подхода является то, что увеличивается объем входного файла и усложняется процесс его подготовки, но описанные выше преимущества значительно превалируют над недостатками.

3.1.2.1 Определение структуры XML-документа и грамматики описания данных

При создании валидных XML-документов необходимо сначала описать их структуру в описателе типа документа (DTD – Document Type Definitions), а затем создавать сами документы, которые должны отвечать всем спецификациям, содержащимся в DTD. Это необходимо для того, чтобы все документы отвечали заданной в DTD-структуре. XML-процессор (например, Microsoft Internet Explorer) может проверить соответствие вновь созданного

документа DTD-структуре. Другими словами, DTD обеспечивает стандартный шаблон для процессора, чтобы при проверке валидности он мог следовать требуемой структуре и гарантировать, что документ соответствует установленным правилам. Если какая-либо часть документа не отвечает DTD-спецификации, процессор выводит сообщение об ошибках. Данный подход, заключающийся в использовании валидных документов, особенно полезен для проверки однородности среди группы схожих документов, ведь для каждого метода должна создаваться собственная спецификация предметной области, описанная в XML-документе. Фактически стандарт XML определяет DTD как "грамматику для определенного класса документов". DTD устанавливает и определяет имена элементов, которые могут быть использованы в документе, порядок, в котором элементы могут появляться, доступные к применению атрибуты элементов и другие особенности документа. Наличие DTD ограничивает круг элементов и структур, которые предполагается использовать, вследствие чего создаваемый документ будет отвечать стандартам разрабатываемого приложения.

Для описания структуры документов, общей для всех методов естественно-языковой обработки научного текста, создадим внешнее DTD, а уникальные отличия каждого метода будем отражать во внутреннем DTD.

Итак, внешнее и внутреннее DTD предназначены для абстрактного описания методов, позволяющих задать грамматику для описания спецификации методов.

В предыдущем разделе был сделан вывод о том, какие структуры данных должны войти в спецификацию методов. Они и должны быть описаны в DTD в абстрактном виде, используя последовательную и выборочную форму модели содержимого элементов.

Описание любого метода состоит из пяти основных элементов:

– объекта исследования (*RESEARCH_OBJECT*);

– множества конструктов (*SET_CONSTRUCTS*);

– альтернативных кортежей вершин первого уровня дерева (*LEVEL_1_TREE_TUPLE*);

– множества вариантов содержимого веток дерева (*SET_BRANCH_TREE*);

– множества понятий и их значений (*SET_VALUE_VERTEX*).

Структура первых четырех основных элементов не меняется, поэтому будем задавать ее во внешнем DTD (рис. 3.3), а структура пятого элемента зависит от конкретного метода, поэтому ее приведем во внутреннем DTD (рис. 3.4).

```
<!ELEMENT METHOD (RESEARCH_OBJECT, CONSTRUCT,
LEVEL_1_TREE_TUPLE,BRANCH_TREE,SET_VALUE_VERTEX)>
<!ATTLIST METHOD NAME CDATA #REQUIRED >
   <!ELEMENT RESEARCH_OBJECT (#PCDATA)> <!— корень дерева продукции -->
   <!ELEMENT SET_CONSTRUCTS (CONSTRUCT+)>
      <!ELEMENT CONSTRUCT (ARC, VERTEX+)>
        <!ATTLIST CONSTRUCT NUMBER CDATA #REQUIRED>
        <!ELEMENT ARC EMPTY>
           <!ATTLIST ARC
              TERM  CDATA "Семантическое отношение"
              VALUE CDATA #REQUIRED     <!— глагол или глагольная группа -->
           >
        <!ELEMENT VERTEX EMPTY>
           <!ATTLIST VERTEX
              TYPE      CDATA #REQUIRED <!— левая или правая вершины -->
              NAMESET   CDATA #REQUIRED <!— имя множества, которомуєтерм -->
              VALUE     CDATA #REQUIRED <!— значение терма в вершине -->
           >
   <!ELEMENT LEVEL_1_TREE_TUPLE (LIST_VERTEX)+>
   <!— альтернативных списков вершины м.б.несколько -->
      <!ELEMENT LIST_VERTEX (ROOT_BRANCH+)> <!— вершины 1-го уровня дерева-->
                        <!— продукции, являющиеся корнями веток дерева -->
      <!ATTLIST LIST_VERTEX NUMBER_VARIANT CDATA #REQUIRED>
        <!ELEMENT ROOT_BRANCH (#PCDATA)>
        <!ATTLIST ROOT_BRANCH CODE_ROOT ID #REQUIRED>
        <!— например,у корня ROOT_BRANCH="КСС" код CODE_ROOT м.б.="CRB01"-->
   <!ELEMENT SET_BRANCH_TREE (BRANCH_TREE+)>
      <!ELEMENT BRANCH_TREE (ROOT* | VERTEX*)+>
      <!— ветвь может иметь вложенные ветви и вершины или -->
      <!— только ветви, или только вершины -->
      <!ATTLIST BRANCH_TREE CODE_ROOT ID #REQUIRED BRANCH_ROOT IDREF
#IMPLIED>
      <!— CODE_ROOT содержит код корня ветки, ветка связывается -->
      <!— со своим корнем через BRANCH_ROOT -->
        <!ELEMENT ROOT (ROOT* | VERTEX*)+>
        <!— т.о. задали рекурсию для возможности формирования -->
        <!— ветвистого дерева -->
        <!ATTLIST ROOT VERTEX CDATA #REQUIRED>
      <!ELEMENT VERTEX (#PCDATA)>
```

Рисунок 3.3 – Внешнее DTD

При описании элементов используются символы '+' и '*', которые означают:

'+' – один или более из предшествующих элементов;

'*' – ни одного или более из предшествующих элементов.

Внешнее DTD должно храниться в отдельном файле «*ConfigSA.dtd*».

Наиболее сложную структуру во внешнем DTD имеет элемент «**BRANCH_TREE**». Это связано с тем, что при описании веток *branc_*? стало ясно, что ветка может иметь различный вид от сильно ветвистого дерева до вложенных веток, содержащих простую последовательность вершин, или даже до одной вершины. Поэтому при описании этого элемента использована выборочная форма модели содержимого элемента с использованием рекурсивного определения вложенного элемента «ROOT».

Во внутреннем DTD описано множество понятий метода и семейств его подмножеств (возможно, пересекающихся), сформированных по отношению принадлежности к множеству понятий верхнего уровня иерархии понятий.

```
<!DOCTYPE METHOD SYSTEM "ConfigSA.dtd"
[
   <!ELEMENT METHOD_WP (SET_VALUE_VERTEX)>
      <!ELEMENT SET_VALUE_VERTEX (LIST_TERM+, VALUE_TERM*)+>
         <!ELEMENT LIST_TERM (TERM+)>
            <!ATTLIST LIST_TERM NAME CDATA #REQUIRED>
            <!ELEMENT TERM EMPTY>
            <!ATTLIST TERM VALUE CDATA #REQUIRED SIGN CDATA #REQUIRED>
         <!ELEMENT VALUE_TERM (VALUE+)>
            <!ATTLIST VALUE_TERM VALUE CDATA #REQUIRED>
]>
```

Рисунок 3.4 – Внутреннее DTD

В описание элемента множества понятий входят имя и знак элемента, его значение, если он им обладает. Например, в методе «Извлечение знаний о семантическом отношении «Целое-часть»» к таким понятиям относятся Признак, Терм-Признаки, Глагол, ПризнакЧасть Речи, Падеж.

3.1.2.2 Пример XML-документа

В данном разделе приведен фрагмент XML-описания спецификации метода «Извлечение знаний о семантическом отношении «Целое-часть»», полный вариант описания которого приведен в Приложении Г. Описание создается модулем «Интерфейс пользователя» на основе введенных данных.

< Фрагмент XML-описания спецификации метода >

```
<?xml version="1.0"?>
<!-- Спецификация метода выявления отношения «Целое-часть». -->
<!-- Файл ConfigSA_MWP.xml -->
<!DOCTYPE METHOD SYSTEM "ConfigSA.dtd"
[
   <!ELEMENT METHOD_WP (SET_VALUE_VERTEX)>
        <!ELEMENT SET_VALUE_VERTEX (LIST_TERM+, VALUE_TERM*)+>
            <!ELEMENT LIST_TERM (TERM+)>
                <!ATTLIST LIST_TERM NAME CDATA #REQUIRED>
                <!ELEMENT TERM EMPTY>
                <!ATTLIST TERM VALUE CDATA #REQUIRED SIGN CDATA #REQUIRED>
            <!ELEMENT VALUE_TERM (VALUE+)>
            <!ATTLIST VALUE_TERM VALUE CDATA #REQUIRED>
]
>
<METHOD NAME="Method_WP">
   <RESEARCH_OBJECT> Предложение </RESEARCH_OBJECT>
   <SET_CONSTRUCTS>
        <CONSTRUCT NUMBER="1">
            <ARC TERM="Семантическое отношение" VALUE="содержит"/>
            <VERTEX TYPE="left"  NAMESET="Предложение" VALUE="Предложение"/>
            <VERTEX TYPE="right" NAMESET="A" VALUE="КСС"/>
            <VERTEX TYPE="right" NAMESET="A" VALUE="ИСС"/>
            <VERTEX TYPE="right" NAMESET="A" VALUE="СемОтношение"/>
            <VERTEX TYPE="right" NAMESET="A" VALUE="ЗаголовТермин"/>
            <VERTEX TYPE="right" NAMESET="A" VALUE="Лексема"/>
            <VERTEX TYPE="right" NAMESET="A" VALUE="Признак"/>
        </CONSTRUCT>
        …
        <CONSTRUCT NUMBER="13">
            <ARC TERM="Семантическое отношение" VALUE="эквивалентен"/>
            <VERTEX TYPE="left"  NAMESET="J" VALUE="Индекс"/>
```

```
          <VERTEX TYPE="right" NAMESET="J" VALUE="Индекс"/>
     </CONSTRUCT>
</SET_CONSTRUCTS>
<!-- Описание вершин первого уровня дерева -->
<LEVEL_1_TREE>
     <LIST_VERTEX NUMBER_VARIANT="1">
          <ROOT_BRANCH CODE_ROOT="v01"> КСС </ROOT_BRANCH>
          <ROOT_BRANCH CODE_ROOT="v02"> СемОтношение </ROOT_BRANCH>
          <ROOT_BRANCH CODE_ROOT="v03"> ИСС </ROOT_BRANCH>
     </LIST_VERTEX>
     …
     <LIST_VERTEX NUMBER_VARIANT="11">
          <ROOT_BRANCH CODE_ROOT="v03"> ИСС </ROOT_BRANCH>
          <ROOT_BRANCH CODE_ROOT="v02"> СемОтношение </ROOT_BRANCH>
          <ROOT_BRANCH CODE_ROOT="v04"> Список  </ROOT_BRANCH>
          <ROOT_BRANCH CODE_ROOT="v08"> Признак </ROOT_BRANCH>
     </LIST_VERTEX>
</LEVEL_1_TREE>
<!-- Описание ветвей дерева -->
<SET_BRANCH_TREE>
     <!-- Описание вариантов ветви с корнем "КСС" -->
     <BRANCH_TREE CODE_ROOT="v01" BRANCH_ROOT="v01">
          <ROOT VERTEX="КСС">
               <ROOT VERTEX="ИСС">
                    <ROOT VERTEX="Терм-спутникХ">
                         <ROOT VERTEX="Значение"> </ROOT>
                    </ROOT>
               </ROOT>
               <ROOT VERTEX="ИСС">
                    <ROOT VERTEX="ЗаголовТермин">
                         <ROOT VERTEX="Характеристика">
                              <ROOT VERTEX="Падеж">
                                   <ROOT VERTEX="Значение"> </ROOT>
                              </ROOT>
                         </ROOT>
                    </ROOT>
               </ROOT>
          </ROOT>
          <ROOT VERTEX="КСС">
               <ROOT VERTEX="ИСС">
                    <ROOT VERTEX="Характеристика">
                         <ROOT VERTEX="Падеж">
                              <ROOT VERTEX="Значение"> </ROOT>
                         </ROOT>
                    </ROOT>
               </ROOT>
               <ROOT VERTEX="ИСС">
                    <ROOT VERTEX="ЗаголовТермин">
                         <ROOT VERTEX="Характеристика">
                              <ROOT VERTEX="Падеж">
                                   <ROOT VERTEX="Значение"> </ROOT>
                              </ROOT>
                         </ROOT>
                    </ROOT>
               </ROOT>
          </ROOT>
          <ROOT VERTEX="КСС">
               <ROOT VERTEX="ИСС">
                    <ROOT VERTEX="Терм-спутникХ">
                         <ROOT VERTEX="Значение">
                         </ROOT>
                         <ROOT VERTEX="Характеристика">
                              <ROOT VERTEX="Падеж">
                                   <ROOT VERTEX="Значение"> </ROOT>
```

```
                                        </ROOT>
                                    </ROOT>
                                </ROOT>
                            </ROOT>
                            <ROOT VERTEX="ИСС">
                                <ROOT VERTEX="ЗаголовТермин">
                                    <ROOT VERTEX="Характеристика">
                                        <ROOT VERTEX="Падеж">
                                            <ROOT VERTEX="Значение"> </ROOT>
                                        </ROOT>
                                    </ROOT>
                                </ROOT>
                            </ROOT>
                            <ROOT VERTEX="Индекс"> </ROOT>
                        </ROOT>
                    </BRANCH_TREE>
                    …
                    <!-- Описание вариантов ветви с корнем "Признак" -->
                    <BRANCH_TREE CODE_ROOT="v08" BRANCH_ROOT="v08">
                        <ROOT VERTEX="Признак">
                            <ROOT VERTEX="Значение"> </ROOT>
                        </ROOT>
                        <ROOT VERTEX="Признак">
                            <ROOT VERTEX="Значение"> </ROOT>
                            <ROOT VERTEX="Индекс"> </ROOT>
                        </ROOT>
                    </BRANCH_TREE>
                </SET_BRANCH_TREE>
                <!-- Описание множеств допустимых значений -->
                <SET_VALUE_VERTEX>
                    <LIST_TERM NAME="A">
                        <TERM VALUE="КСС" SIGN="k"/>
                        <TERM VALUE="ИСС" SIGN="s" />
                        <TERM VALUE="СемОтношение"  SIGN="r"/>
                        <TERM VALUE="ЗаголовТермин" SIGN="z"/>
                        <TERM VALUE="Термин"  SIGN="t"/>
                        <TERM VALUE="Лексема" SIGN="l"/>
                        <TERM VALUE="Список"  SIGN="q"/>
                    </LIST_TERM>
                    …
                    <LIST_TERM NAME="J">
                        <TERM VALUE="Индекс" SIGN="i"/>
                        <TERM VALUE="Падеж"  SIGN="c"/>
                        <TERM VALUE="ЧастьРечи" SIGN="c"/>
                    </LIST_TERM>
                    <VALUE_TERM VALUE="Падеж">
                        <VALUE> Им  </VALUE>
                        <VALUE> Род </VALUE>
                        <VALUE> Вин </VALUE>
                        <VALUE> Тв  </VALUE>
                    </VALUE_TERM>
                    …
                    <VALUE_TERM VALUE="Признак">
                        <VALUE> : </VALUE>
                    </VALUE_TERM>
                </SET_VALUE_VERTEX>
            </METHOD>
```

Подобные описания используются в генераторе систем продукций для настройки заданного метода на генерацию продукционных правил.

3.2 Генетическое программирование в решении задачи генерации ядер продукционных правил

Данный раздел посвящен решению проблемы автоматического построения продукционных правил на основе применения генетического программирования [24,42,104].

3.2.1 Основные положения генетического алгоритма

Продукционная модель или модель, основанная на правилах, позволяет представить знания в виде утверждений, имеющих вид предложений естественного языка типа: «Если A, то B». Под антецедентом A и консеквентом B понимается некоторое множество фактов. Каждый факт имеет структуру простой ядерной конструкции языка ситуационного моделирования xRy, где x, y – термины, R – семантическое отношение.

На основе XML-описания спецификации метода необходимо сформировать допустимое множество ядер продукций NPK (Nature Production Kernel) в виде утверждений на ограниченном подмножестве естественного языка при следующих ограничениях: утверждение должно содержать не менее 2 и не более 30 фактов; факты, составляющие утверждение, должны быть корректными.

Для решения поставленной задачи используем методологию генетического программирования.

В данной работе за основу взяты следующие положения генетического алгоритма.

1. Ядро продукционного правила представляется в виде хромосомы.
2. Хромосома состоит из динамического набора молекул ДНК и представляется в виде дерева.
3. Молекула ДНК состоит из динамического количества генов (минимальная неделимая составная часть молекулы ДНК).
4. Ген является понятием метода.
5. Первая популяция $P(0)$ генерируется случайным образом, при этом значение гена выбирается из входного алфавита (множество понятий метода T_k).
6. Каждое новое поколение $P(t)$ генерируется при помощи оператора кроссинговера. Родительские пары выбираются оператором репродукции различными методами: колеса рулетки, пропорциональной шкалы и турнирного отбора.
7. В каждой новой хромосоме некорректные и повторяющиеся молекулы подвергаются мутации.
8. Все повторяющиеся хромосомы из популяции удаляются.
9. Решением генетического алгоритма является не одна особь, а совокупность особей, удовлетворяющих вышеприведенным требованиям.
10. Оцениваются особи и совокупности особей.

Структура хромосомы имеет вид дерева. Молекула ДНК состоит из динамического количества генов и представляется в виде

$$<\alpha_i \rightarrow \beta_1, lc, \beta_2, lc, \ldots, lc, \beta_m>, \quad (3.4)$$

где $\alpha_i, \beta_1, \beta_2, \ldots, \beta_m, lc$ – гены, $max(m)=7$;

$\alpha_i, \beta_j \in T_k$, T_k – входной алфавит, $|T_k|= k$;

символ '$\rightarrow$' обозначает глагол;

lc – логическая связка «И» или «ИЛИ».

Например, если $\alpha_i= p$, $\beta_1,= k$, $\beta_2 = h$, то граф молекулы ДНК будет иметь вид, изображенный на рисунке 3.5.

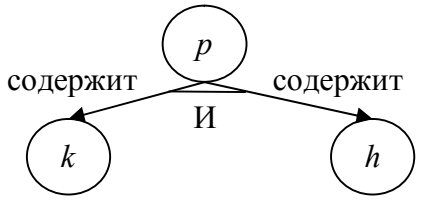


Рисунок 3.5 – Пример молекулы ДНК

Пара смежных веток, принадлежащих одной вершине, связана между собой логической связкой «И» или «ИЛИ». Логическая связка «И» обозначается символом ',', «ИЛИ» – ';'. Глагол может иметь отрицание, то есть унарную логическую связку «¬», которая записывается над стрелкой: $\xrightarrow{\neg}$.

Функция оценки особи строится на основе формулы Дайса [14]. Для оценки совокупности особей используется объект, внешний для генетического алгоритма системы логического резолютивного вывода *LogResDed* (Logical Resolutive Deduction). Эта система определяет приспособленность совокупности особей к решению поставленной прикладной задачи, другими словами, определяет достоверность множества NPK метода.

Fitness-функция F особи вычисляется только для веток, корень которых входит в список вершин первого уровня дерева. Список LEVEL_1_TREE_TUPLE описан в XML-документе метода.

По каждой разрешенной ветке вычисляется Fitness-функция F_i по формуле Дайса:

$$F_i = 2n(G_1 \cap G_2) / (n(G_1) + n(G_2)), \quad (3.5)$$

где G_1 – ветка дерева, описанная в спецификации метода;

G_2 – текущая ветка сгенерированного дерева.

При этом графы G_1 и G_2 имеют одинаковый корень.

Общая формула для Fitness-функции F особи равна:

$$F = \sum_{i=1}^{n} F_i , \quad (3.6)$$

где n – количество разрешенных веток в сгенерированном дереве.

После оценивания выполняются операторы селекции и мутации с целью улучшения особей популяции. Когда функция оценки совокупности особей достигнет единичного значения, то данная особь копируется в конечное множество.

Новая популяция создается посредством использования операторов кроссинговера.

3.2.2 Генетический алгоритм генерации ядер продукционных правил

Для решения задачи был разработан генетический алгоритм, основанный на методологии эволюционного программирования [24,42,134] и исследованиях генетических операторов различных видов [123,135,143]. Цель алгоритма – построить минимальное по мощности множество правил $R_1,\ldots, R_n$, такое, что они, являясь базовыми правилами метода, корректно решают прикладную задачу, соответствующую методу. Схема алгоритма представляется следующим образом.

1. Создание начальной популяции.
2. Первоначальное оценивание, редукция и мутация.
3. Селекция, мутация, скрещивание, редукция.
4. Оценивание особи: если функция оценки i-й особи $F_i^{Chr} = 1$, то переход на пункт 5, иначе – на пункт 3.
5. Копирование особи в результирующее множество.
6. Определение останова: если после n новых поколений новые корректные особи не появились, то переход на пункт 7, иначе – на создание нового поколения – пункт 3.
7. Оценивание совокупности особей с помощью системы *LogResDed*, если Fitness-функция $F(PS) = 1$, то останов, иначе – на пункт 3.

Особью является ядро продукционного правила, представленное множеством ДНК в виде дерева. В соответствии с положениями генетического алгоритма вначале случайным образом создается начальная популяция $P(0)$. При этом необходимо проследить, чтобы начальная популяция обладала набором особей, непосредственно описывающих верхние и нижние граничные значения генов. Это позволит создать разнообразную (с хорошей вариативностью) и качественную (с хорошим фенотипом) популяцию.

Оператор редукции. Текущая популяция подвергается естественному отбору посредством оператора редукции, который реализует три функции:

1) исключение из популяции некорректных особей или ДНК особей, которые могли возникнуть в результате генерации начальной популяции и применения операторов скрещивания и мутации, а также особей, не удовлетворяющих ограничениям задачи;

2) удаление из популяции одинаковых индивидов во избежание вырождения популяции, когда потомки одного индивида с хорошей приспособленностью заполняют популяцию, уменьшая разнообразие генетического материала;

3) ограничение размера популяции в случае, если задан ее максимальный размер, при этом из популяции удаляются особи с наименьшим значением Fitness-функции.

Оператор мутации. Мутации будем подвергать молекулы ДНК хромосомы. Цель мутации – улучшить характеристики ДНК. Основные операции мутации:

– упорядочение ДНК в хромосоме;

– удаление одинаковых молекул ДНК за исключением одной;

– добавление молекул ДНК, например, при отсутствии служебного слова ТО;

– модификация молекул ДНК: изменение левого или правого терм-спутника при их равенстве; изменение одного из термов: если x есть терм-спутник, то y должен быть базовым термом и наоборот; изменение правого терма на терм «глагол», так как, если x есть терм-спутник R, то y должен быть базовым термом «глагол»; и т.п.

Оператор селекции. Цель оператора селекции – выбрать из множества индивидов популяции $P(t)$ пару особей для выполнения оператора кроссинговера. В алгоритме предлагается использовать метод селекции на основе заданной шкалы. Особи PS_i популяции $P(t)$ ранжируются по значению $F_i(PS)$. В соответствии со стратегией перехода из одной подобласти альтернативных решений в другую, повышающей эффективность поиска оптимального решения, назначение значений шкалы будем осуществлять одновременно сверху вниз и снизу вверх, что обеспечит возможность выбора пары, состоящей из лучшей и худшей особей. Таким образом, для скрещивания выбирается пара особей с одинаковыми значениями шкалы.

Оператор кроссинговера. Оператор скрещивания (кроссинговера) используется для получения новых особей на базе уже имеющихся. Для скрещивания используются двух-, трех- и четырехточечные операторы кроссинговера.

3.2.3 Функция приспособленности совокупности особей

Цель вычисления фитнесс-функции совокупности особей заключается в оценивании корректности работы метода, представленного в виде множества продукционных правил. Для этого сгенерированная и отобранная совокупность особей (модели ядер продукционных правил) подвергается ряду отображений:

– особь как модель утверждения (ядро продукционного правила) из десятичного представления – в естественно-языковое;

– утверждение на естественном языке – на язык логики предикатов первого порядка;

– формула на языке логики предикатов – во множество дизъюнктов.

Интерпретация сгенерированной особи в продукционное правило на ограниченном подмножестве естественного языка должна удовлетворять следующим требованиям:

1) термы должны быть заключены в угловые скобки < >;

2) семантические отношения должны быть заключены в фигурные скобки {};

3) значения должны быть заключены в квадратные скобки [ ];

4) в утверждении после терма должно следовать его обозначение;

5) выражение должно записываться в круглых скобках;

6) служебные слова ЕСЛИ, ТО и логические связки И, ИЛИ должны записываться без кавычек;

7) термы должны быть нормализованы, то есть находиться в именительном падеже и единственном числе;

8) вектор должен записываться как функция.

После этого к ядру продукции добавляются остальные компоненты продукционного правила. Сформированная система продукций подается на вход аппарата активации продукционных правил, основой которого является система логического резолютивного вывода *LodResDed*.

Если в 95% тестовых примеров система продукций решает поставленную прикладную задачу, то считается, что проблема генерации системы продукций завершена, иначе генерация должна быть продолжена.

К настоящему времени сгенерированы системы продукций по методам, которые реально используются в лабораторных экспериментах. Продолжается процесс генерации систем продукций по другим методам.

Таким образом, описанный в работе способ генерации множества ядер продукций в конечном итоге позволит сформировать библиотеку декларативных методов естественно-языковой обработки научных текстов, которая может быть использована для автоматического построения онтологий предметных областей.

3.3 Преобразование продукционных правил

Продукционные правила, генерация которых описана в предыдущем разделе, необходимо преобразовать в формальное представление. Для этого необходимо решить две задачи:

1) преобразование утверждения на естественном языке – на язык логики предикатов первого порядка;

2) преобразование формулы на языке логики предикатов – во множество дизъюнктов.

Решение этих задач связано с созданием модели преобразователя и её реализацией. На основе анализа трудов [28,87,125,126,131,136,144] в работе создана модель с применением технологии генетического программирования, а для реализации модели использованы технологии автоматного программирования.

3.3.1 Обобщенная схема генерации модели автоматического преобразователя

Генератор модели автоматического преобразователя должен работать в тесном контакте с инструментальной системой UniMod [94], поддерживающей технологии автоматного программирования. Это объясняется тем, что функция пригодности особей сгенерированной популяции, прошедшей предварительный отбор, вычисляется на основе

результатов, полученных после прогонки каждой особи в среде UniMod. Принципиальную схему работы генератора (рис. 3.6) можно представить в виде следующей последовательности действий.

1. Осуществляется выбор прикладной задачи, для решения которой следует сгенерировать модель преобразователя с помощью генетического алгоритма, вводятся данные предметной области задачи. На основе этих данных создается спецификация задачи в виде XML-описания и записывается в файле с идентификатором "*config.xml*".

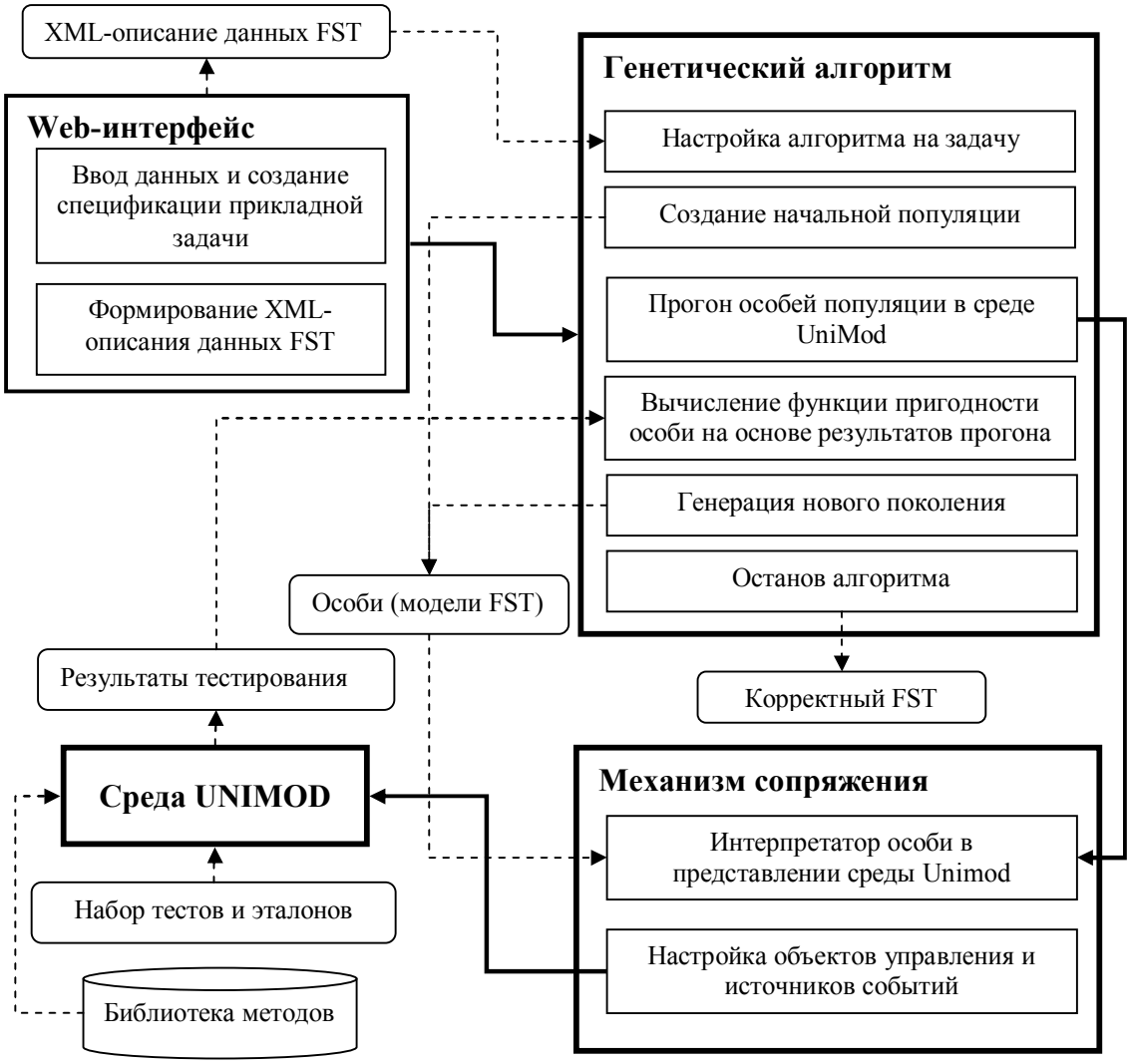


Рис. 3.6 – Обобщенная схема модели генератора

2. Создается новая популяция размером, заданным в конфигурационном файле.

3. Выполняется тестирование каждой особи в среде UniMod на тестовых данных. Для прогонки особи в среде UniMod генерируется XML-описание модели автомата. По результатам тестирования вычисляется оценка (Fitness-функция) данного автомата.

4. Проверяется значение Fitness-функции: если оно достигло своего максимального значения, заданного в файле "*config.xml*", то осуществляется переход на пункт 6, иначе – на следующий пункт.

5. Генерируется новое поколение, затем управление передается на пункт 3.

6. Формирование исполняемого кода преобразователя по лучшей модели *FST*, останов алгоритма.

В соответствии со схемой, приведенной на рисунке 3.6, модель генератора автоматического преобразователя формально определяется тройкой [47,68,72,74,78]:

$$M = < GenUI, GA, IT>, \quad (3.7)$$

где *GenUI* – пользовательский интерфейс;

GA – компонент генерации модели преобразователя;

IT – транслятор для сопряжения со средой UniMod.

Рассмотрим суть каждого компонента.

3.3.2 Интерфейс пользователя GenUI

Трёхкомпонентный интерфейс пользователя представляет собой графический интерфейс и имеет следующее формальное определение:

$$GenUI = <CI, XML\_B, DBM >, \quad (3.8)$$

где *CI* (Conversational Input) – компонент, отвечающий за ввод исходной информации в соответствии с выставленным шаблоном и контроль введенной информации;

XML_B (XML Building) – компонент, обеспечивающий формирование XML-описания исходных данных о модели по введенным данным об автомате;

DBM – компонент, отвечающий за ведение базы данных *ApplArea*.

Компонент *CI* обеспечивает ввод и обновление данных о предметной области решаемого класса задач: входных и выходных переменных, входных и выходных потоков, ввод исходных данных для запуска автомата.

Компонент *XML_B* отвечает за формирование XML-описания исходных данных о модели преобразователя.

Компонент *DBM* предназначен для поддержки работы с базой данных *ApplArea*. База данных состоит из несвязанных таблиц, которые необходимы для различных прикладных задач. Так, при генерации модели преобразователя классических продукций используется таблица *Relation*, а нечетких продукций – таблица *Fuzzy*. Эти таблицы используются при распознавании семантического отношения.

Взаимодействие с базой данных *dbRelation* реализовано стандартными методами посредством *ODBC* (Open DataBase Connectivity). Для этого используется источник данных *myConnection*.

Результатом работы интерфейсной части являются спецификация данных о предметной области задачи, записанная в базу данных *ApplArea*, и конфигурационный файл (XML-описание) "*ConfigTask.xml*", содержащий необходимые исходные данные для генерации автомата.

Рассмотрим вопрос формирования файла “*ConfigTask.xml*”.

Так как в данном случае генерируется модель преобразователя, основными входными данными модели являются входной и выходной алфавит, состояния и действия преобразователя. Назовем эти элементы “ALPHALIST”, “BETALIST”, “STATELIST” и “ACTLIST”, соответственно. В связи с тем, что в каждом состоянии выполняется только одно действие, то списки “STATELIST” и “ACTLIST” совпадают. Введем вспомогательные элементы, описывающие параметры генетического алгоритма.

CICLECOUNT – максимальное количество входов в одно состояние, при достижении которого автомат прерывает своё выполнение. Служит для обхода ошибки от зацикливания автоматов на этапе эволюции.

DELTA – максимальное значение Fitness-функции, при достижении которой эволюция прекращается, то есть является условием выхода из цикла эволюции.

POPSIZE – размер популяции.

PARPOOL – нормализованная величина родительского пула.

MUTPOOL – нормализованная величина мутантов по отношению к родительскому пулу.

TOURNUM – допустимое количество хромосом, участвующих в одном турнирном соревновании при использовании турнирной селекции.

ROULNUM – допустимое количество хромосом, участвующих в одном розыгрыше рулетки при использовании селекции методом рулетки.

SELTYPE – тип селекции. Указывается тип или типы селекии, которые могут использоваться в генетическом алгоритме. Тип выбирается из следующего списка: турнирный отбор, колесо рулетки и пропорциональная селекция.

Кроме того, должны задаваться следующие идентификаторы файлов.

ETALONNAME – элемент, содержащий идентификаторы файлов с эталонными продукциями.

INPUTNAME – элемент, содержащий идентификаторы файлов с продукциями на естественном языке, подлежащими преобразованию.

OUTPUTNAME – элемент, содержащий идентификаторы файлов с преобразованными продукциями, представленными на языке логики предикатов первого порядка.

Введем вспомогательные элементы, описывающие требуемые параметры предметной области прикладной задачи.

STANDARTLIB – стандартные библиотеки действий.

USERLIB – пользовательские библиотеки действий.

ARITY – максимальная арность предиката.

TYPEOA – тип объекта анализа, например, отношение «Целое-часть» или «Дефиниция».

В связи с тем, что все вышеперечисленные элементы одинаковы для всех генерируемых преобразователей, их можно описать во внешнем DTD (рис. 3.7) с идентификатором “*ConfigTask.dtd*”.

```
<!ELEMENT TASK (ALPHABET, LIST_FILE, PARAMETRES_GA)>
<!ATTLIST TASK NAME CDATA #REQUIRED >
   <!– алфавиты -->
   <!ELEMENT ALPHABET (ALPHALIST, BETALIST)>
        <!ELEMENT ALPHALIST (SYMBOL+)>
                <!ELEMENT SYMBOL EMPTY>
                <!ATTLIST SYMBOL NAME CDATA #REQUIRED>
        <!ELEMENT BETALIST (SYMBOL+)>
                <!ELEMENT SYMBOL EMPTY>
                <!ATTLIST SYMBOL NAME CDATA #REQUIRED>
   <!– состояния и действия -->
   <!ELEMENT STATE_ACT (ACTLIST)>
        <!ELEMENT ACTLIST (SYMBOL+)>
                <!ELEMENT SYMBOL EMPTY>
                <!ATTLIST SYMBOL NAME CDATA #REQUIRED>
   <!– списки файлов -->
   <!ELEMENT LIST_FILE (ETALONNAME, INPUTNAME, OUTPUTNAME)>
        <!ELEMENT ETALONNAME (FILE+)>
                <!ELEMENT FILE EMPTY>
                <!ATTLIST FILE NAME CDATA #REQUIRED>
        <!ELEMENT INPUTNAME (FILE+)>
                <!ELEMENT FILE EMPTY>
                <!ATTLIST FILE NAME CDATA #REQUIRED>
        <!ELEMENT OUTPUTNAME (FILE+)>
                <!ELEMENT FILE EMPTY>
                <!ATTLIST FILE NAME CDATA #REQUIRED>
   <!– параметры генетического алгоритма -->
   <!ELEMENT PARAMETRES_GA (CICLECOUNT, DELTA, POPSIZE, PARPOOL, MUTPOOL,
              TOURNUM, ROULNUM, SELTYPE)>
        <!ELEMENT CICLECOUNT (#PCDATA)>
        <!ELEMENT DELTA (#PCDATA)>
        <!ELEMENT POPSIZE (#PCDATA)>
        <!ELEMENT PARPOOL (#PCDATA)>
        <!ELEMENT MUTPOOL (#PCDATA)>
        <!ELEMENT TOURNUM (#PCDATA)>
        <!ELEMENT ROULNUM (#PCDATA)>
      <!ELEMENT SELTYPE (#PCDATA)>
   <!– библиотеки -->
   <!ELEMENT LIBRARY_LIST (STANDARTLIB, USERLIB)>
        <!ELEMENT STANDARTLIB (LIBRARY+)>
                <!ELEMENT LIBRARY EMPTY>
                <!ATTLIST LIBRARY NAME CDATA #REQUIRED>
        <!ELEMENT USERLIB (LIBRARY +)>
                <!ELEMENT LIBRARY EMPTY>
                <!ATTLIST LIBRARY NAME CDATA #REQUIRED>
   <!– параметры предметной области -->
   <!ELEMENT PARAMETRES_SA (ARITY, TYPEOA)>
        <!ELEMENT ARITY (#PCDATA)>
        <!ELEMENT TYPEOA (#PCDATA)>
```

Рисунок 3.7 – Внешнее DTD

Пример файла “*ConfigTask.xml*” приведен в приложении Д. Разработаем структуру XML-документов для входных и выходных файлов задачи. К входным файлам относятся файлы, содержащие преобразуемые и эталонные продукции, выходным – файлы, содержащие преобразованные продукции. XML-документы, описывающие данные файлы, имеют одинаковую достаточно простую структуру. Так как эти файлы являются аналогом входных и выходных потоков преобразователя, то все элементы документа (за исключением корневого) будут иметь имя «symbol». Для различения

элементов введем обязательный атрибут «type». Этот атрибут будет характеризовать контекст использования элемента.

В документе, содержащем входную продукцию на ограниченном подмножестве естественного языка, один элемент «symbol» представляет собой одну ячейку входной ленты автомата, которая считывается и обрабатывается как единое целое. Обозначим идентификатор XML-документа входного файла через «*Input#.xml*», идентификатор XML-документа с эталонной продукцией – «*Etalon#.xml*», идентификатор XML-документа выходного файла – «*Output#.xml*». В идентификаторах символ '#' обозначает номер файла. Выходная (преобразованная) и эталонная продукции представлены на языке логики предикатов первого порядка, поэтому их структуры абсолютно идентичны. Рассмотрим структуру XML-документа «*Input.xml*».

Определим четыре типа (TYPE) элементов SYMBOL:

1) оператор (operator);

2) левый терм (*x*) простой ядерной конструкции *xRy*;

3) правый терм (*y*) простой ядерной конструкции *xRy*;

4) семантическое отношение (*R*) простой ядерной конструкции *xRy*.

К операторам относятся элементы, представляющие «служебные» слова, образующие конструкцию продукции. Все четыре элемента имеют, кроме атрибута «TYPE», атрибут «VALUE». Так, например, при TYPE="operator" атрибут «VALUE» отражает название и смысл оператора и принимает следующие значения: «if», «then», «or», «and», «eof».

Элементы типа TYPE="*x*" или TYPE="*y*" содержат вложенные элементы со своими атрибутами «TYPE» и «VALUE». В данных элементах атрибут «TYPE» принимает два значения:

1) терм метода (*term*);

2) обозначение терма (*sign*).

Атрибут «VALUE» при TYPE="*term*" содержит значение одного из имеющихся в базе знаний термов, а при TYPE="*sign*" – обозначение терма.

Элементы типа TYPE="*R*" имеют атрибут «TYPE», принимающий значение «semrel» и атрибут «VALUE», содержащий глагол, определяющий семантическое отношение между термами *x* и *y*, например, «содержит», «имеет», «есть» и т.п.

Представленный ниже фрагмент XML-документа описывает пример простой ядерной конструкции вида «предложение p_1 (*x*) содержит (*R*) термин *z* (*y*)»:

```
<SYMBOL TYPE="y">
  <SYMBOL TYPE="term" VALUE="термин"/>
  <SYMBOL TYPE="sign" VALUE="z"/>
</SYMBOL>
<SYMBOL TYPE="x">
  <SYMBOL TYPE="term" VALUE="предложение"/>
  <SYMBOL TYPE="sign" VALUE="p1"/>
</SYMBOL>
<SYMBOL TYPE="r">
  <SYMBOL TYPE="semrel" VALUE="содержит"/>
</SYMBOL>.
```

Структура XML-документа «*Input.xml*» в общем виде может быть представлена следующим образом:

```
<INPUT>
  <SYMBOL TYPE="operator" VALUE="if"/>
  <SYMBOL TYPE="y">
    <SYMBOL TYPE="term" VALUE="…"/>
    <SYMBOL TYPE="sign" VALUE="…"/>
  </SYMBOL>
  <SYMBOL TYPE="x">
    <SYMBOL TYPE="term" VALUE="…"/>
    <SYMBOL TYPE="sign" VALUE="…"/>
  </SYMBOL>
  <SYMBOL TYPE="r">
    <SYMBOL TYPE="semrel" VALUE="…"/>
  </SYMBOL>
  [<SYMBOL TYPE="operator" VALUE="or" | "and"/>
  <SYMBOL TYPE="y">
    <SYMBOL TYPE="term" VALUE="…"/>
    <SYMBOL TYPE="sign" VALUE="…"/>
  </SYMBOL>
  <SYMBOL TYPE="x">
    <SYMBOL TYPE="term" VALUE="…"/>
    <SYMBOL TYPE="sign" VALUE="…"/>
  </SYMBOL>
  <SYMBOL TYPE="r">
    <SYMBOL TYPE="semrel" VALUE="…"/>
  </SYMBOL>]*

  …
  <SYMBOL TYPE="operator" VALUE="then"/>
  <SYMBOL TYPE="y">
    <SYMBOL TYPE="term" VALUE="…"/>
    <SYMBOL TYPE="sign" VALUE="…"/>
  </SYMBOL>
  <SYMBOL TYPE="x">
    <SYMBOL TYPE="term" VALUE="…"/>
    <SYMBOL TYPE="sign" VALUE="…"/>
  </SYMBOL>
  <SYMBOL TYPE="r">
    <SYMBOL TYPE="semrel" VALUE="…"/>
  </SYMBOL>
  [<SYMBOL TYPE="operator" VALUE="or | and"/>
  <SYMBOL TYPE="y">
    <SYMBOL TYPE="term" VALUE="…"/>
    <SYMBOL TYPE="sign" VALUE="…"/>
  </SYMBOL>
  <SYMBOL TYPE="x">
    <SYMBOL TYPE="term" VALUE="…"/>
    <SYMBOL TYPE="sign" VALUE="…"/>
  </SYMBOL>
  <SYMBOL TYPE="r">
    <SYMBOL TYPE="semrel" VALUE="…"/>
  </SYMBOL>]*
  <SYMBOL TYPE="operator" VALUE="eof"/>
</INPUT>.
```

В квадратных скобках указаны необязательные элементы. Звездочка «*» после элемента означает, что он может быть повторен произвольное количество раз (включая 0). Символом «…» обозначены произвольные значения атрибутов.

Пример XML-документа «*Input.xml*» приведен в приложении Е.

Структура XML-документов «*Etalon#.xml*» и «*Output#.xml*» с эталонной и выходной продукциями соответственно содержит два основных типа (TYPE) элементов SYMBOL:

1) оператор (operator);

2) пропозициональный символ (*PS*) предиката.

Элементы типа TYPE="*PS*", в свою очередь, содержат вложенные элементы с атрибутами «TYPE» трех типов:

1) первый аргумент (*FArg*) предиката;

2) второй аргумент (*x*) предиката;

3) третий аргумент (*y*) предиката.

Все перечисленные типы элементов имеют второй атрибут «VALUE», область значений которого определяется значением TYPE. При TYPE="operator" атрибут «VALUE» принимает значения «if», «then», «or», «and», «eof».

При TYPE="*PS*" атрибут «VALUE» содержит предикатный символ, соответствующий конкретному семантическому отношению. Например, *PHier* – предикатный символ, соответствующий квалитативному отношению иерархии, относящемуся к понятийной сфере «абстрактное – конкретное», *PAggr* – предикатный символ, соответствующий квалитативному отношению агрегации, относящемуся к понятийной сфере принадлежности.

При TYPE="*FArg*" атрибут «VALUE» содержит первый аргумент предиката, уточняющий отношение, заданное предикатным символом. Например, VALUE="*Value*" уточняет отношение иерархии и идентифицирует отношение «Признак↔значение признака», VALUE="Whole" уточняет отношение агрегации и идентифицирует отношение «Целое↔часть» и т.п.

Атрибуты «VALUE» второго и третьего аргументов типа TYPE="*x*" или TYPE="*y*" соответствуют их смысловому именованию и содержат обозначения термов.

Продемонстрируем создание XML-документа «*Etalon#.xml*» на примере описания утверждения «предложение p_1 (*x*) содержит (*R*) термин *z* (*y*)», которое на языке логики предикатов первого порядка запишется в виде предиката *PAggr*(Whole, p_1, *z*). В соответствии с описанной выше структурой фрагмент XML-документа, описывающий данный предикат, будет иметь вид:

```
<SYMBOL TYPE="PS" VALUE="PAggr"/>
  <SYMBOL TYPE="FArg" VALUE="whole"/>
  <SYMBOL TYPE="x" VALUE="p1"/>
  <SYMBOL TYPE="y" VALUE="z"/>
</SYMBOL>.
```

Таким образом, структура XML-документа «*Etalon#.xml*» в общем виде будет иметь вид:

```
<ETALON>
  <SYMBOL TYPE="operator" VALUE="if"/>
  <SYMBOL TYPE="PS" VALUE="…"/>
    <SYMBOL TYPE="FArg" VALUE="…"/>
    <SYMBOL TYPE="x" VALUE="…"/>
    <SYMBOL TYPE="y" VALUE="…"/>
  </SYMBOL>
  [<SYMBOL TYPE="operator" VALUE="or | and"/>
```

```
    <SYMBOL TYPE="PS" VALUE="…"/>
      <SYMBOL TYPE="FArg" VALUE="…"/>
      <SYMBOL TYPE="x" VALUE="…"/>
      <SYMBOL TYPE="y" VALUE="…"/>
    </SYMBOL>]*

    …
    <SYMBOL TYPE="operator" VALUE="then"/>
    <SYMBOL TYPE="PS" VALUE="…"/>
      <SYMBOL TYPE="FArg" VALUE="…"/>
      <SYMBOL TYPE="x" VALUE="…"/>
      <SYMBOL TYPE="y" VALUE="…"/>
    </SYMBOL>
    [<SYMBOL TYPE="operator" VALUE="or | and"/>
    <SYMBOL TYPE="PS" VALUE="…"/>
      <SYMBOL TYPE="FArg" VALUE="…"/>
      <SYMBOL TYPE="x" VALUE="…"/>
      <SYMBOL TYPE="y" VALUE="…"/>
    </SYMBOL>]*
    <SYMBOL TYPE="operator" VALUE="eof"/>
  </ETALON>.
```

XML-документ «*Output#.xml*» имеет аналогичную структуру. Пример XML-документа «*Etalon.xml*» приведен в приложении Ж.

3.3.3 Генетический алгоритм генерации модели преобразователя

Генетический алгоритм *GA* (генератор модели преобразователя) представляется трехкомпонентной моделью:

$$GA = < Common,\ Population,\ OG >, \tag{3.8}$$

где *Common* – компонент, обеспечивающий взаимодействие как между компонентами верхнего уровня, так и между компонентами *GA*, а также реализует взаимодействие с базой данных *ApplArea*;

Population – компонент, отвечающий за работу с популяцией: представление популяции, отдельной хромосомы и ее частей (молекул ДНК); рекурсивное создание ДНК и заполнение ими хромосомы; вычисление *Fitness*-функции особи, популяции;

OG – компонент, обеспечивающий реализацию генетических операторов: селекции, кроссинговера и мутации.

3.3.3.1 Основные положения генетического алгоритма

Данный генетический алгоритм [63] основан на следующих положениях.

1. Автомат (особь) представляется в виде хромосомы.
2. Хромосома состоит из динамического набора молекул ДНК и имеет графовое представление.
3. Молекула ДНК состоит из статического количества генов (минимальная неделимая составная часть молекулы ДНК).
4. Первая популяции *P*(0) генерируется случайным образом, при этом значение каждого гена выбирается из соответствующей области допустимых значений, заданной в конфигурационном файле “*configTask.xml*”.

5. Каждое новое поколение $P(t)$ генерируется при помощи одного из трех видов оператора кроссинговера или их комбинации.

6. Родительские пары выбираются оператором репродукции по одному из трех методов.

6. В каждой новой хромосоме повторяющиеся молекулы, если таковые есть, подвергаются мутации.

7. Все повторяющиеся хромосомы из популяции удаляются.

8. Каждая особь каждой популяции прогоняется в среде UniMod на тестовых продукциях, представленных на ограниченном подмножестве естественного языка.

9. Оценка хромосом и их ранжирование осуществляются на основе Fitness-функции, вычисленной по модифицированной формуле Дайса.

Структура хромосомы. Структура хромосомы имеет вид графа и содержит динамический набор двухкомпонентных молекул ДНК, каждый из которых содержит шесть генов:

$$<CurState, NewStateL, \alpha, \beta, 0, Action>, \quad (3.9)$$

$$<CurState, NewStateR, \alpha, \beta, 1, Action>, \quad (3.10)$$

где *CurState* – текущее состояние и $CurState \in$ STATELIST;

NewStateL – новое состояние, в которое переходит автомат в случае, если преобразование $\alpha \rightarrow \beta$ дает ложное значение, то есть при нулевом пятом гене левой молекулы ДНК, $NewStateL \in$ STATELIST;

NewStateR – новое состояние, в которое переходит автомат в случае, если преобразование $\alpha \rightarrow \beta$ дает истинное значение, то есть при единичном пятом гене правой молекулы ДНК, $NewStateR \in$ STATELIST;

Action – действие, которое выполняется в текущем состоянии, $Action \in$ ACTIONLIST;

α – входной символ и $\alpha \in$ ALPHALIST;

β – выходной символ и $\beta \in$ BETALIST.

Оценивание хромосомы. После создания каждой *i*-й популяции заданного размера *popsize* (параметр генетического алгоритма, содержащийся в конфигурационном файле "*ConfigTask.dtd*") выполняется прогон особей в среде UniMod. Для прогона моделей преобразователей в среде UniMod разработаны метод «Интерпретатор хромосомы в представление среды UniMod» и библиотека методов, содержащая функции, реализующие действия преобразователя.

Прогонка модели преобразователя осуществляется на множестве тестовых входных файлов с идентификаторами *InPut#.xml*. После прогона создается выходной файл *OutPut#.xml*, содержащий преобразованную продукцию. Каждому тестовому файлу *InPut#.xml* соответствует эталонный файл *Etalon#.xml*, который используется для расчета *Fitness*-функции особи.

В качестве *Fitness*-функции используется модифицированная функция расчета меры близости графов, основанная на коэффициенте Дайса [14,128,138]. Для двух графов G_E и G_T мера близости F_T вычисляется по формуле:

$$F_T = \frac{\sum_{i=1}^{5} k_i \cdot n(S_i^T \cap S_i^E)}{\sum_{i=1}^{5} k_i \cdot n(S_i^E)}, \qquad (3.11)$$

где $n(S)$ – мощность множества вершин S графов G_E и G_T;

E – индекс эталонного графа, хранящегося в файле *Etalon#.xml*;

T – индекс текущего выходного графа *OutPut#.xml*;

i – индекс вершины: *PS*, *PArg*, *Operator*, *X*, *Y*;

k_i – масштабирующий коэффициент соответствующих вершин.

Все особи ранжируются по значению функции F_T, выбирается первое максимальное значение *Fitness*-функции. Если $F_1 < \Delta$, то генерируется новая популяция, иначе выполнение генетического алгоритма завершается и считается, что поиск модели преобразователя завершен. Значение Δ задано в конфигурационном файле "*ConfigTask.xml*" в виде параметра DELTA.

3.3.3.2 Генетические операторы

Новая популяция формируется на основе операторов кроссинговера и мутации. Количество мутаций MUTPOOL задается в конфигурационном файле. Создание родительского пула (*pool*) для скрещивания или мутации осуществляется в зависимости от значения параметра *seltype* (тип селекции), считанного из конфигурационного файла, одним из трех операторов селекции:

- пропорциональная селекция;
- турнирная селекция;
- колесо рулетки.

Операторы селекции. При *турнирной селекции* случайным образом отбираются из текущей популяции хромосомы, количество которых задано параметром *TOURNUM* конфигурационного файла (*TOURNUM*≥2). Из них выбирается хромосома с максимальным значением *Fitness*-функции, остальные отбрасываются. Этот процесс повторяется q раз, где q = 10% от размера популяции.

Этот метод не обладает преждевременной сходимостью, не имеет стагнации и не требует глобального переупорядочения.

При *селекции методом колеса рулетки* тоже случайным образом отбираются из текущей популяции хромосомы, количество которых задано параметром *ROULNUM*. Колесо рулетки содержит по одному сектору для каждой отобранной хромосомы. Размер i-го сектора пропорционален соответствующей величине $P_{sel}(i)$, вычисляемой по формуле:

$$P_{sel}(i) = f(i) \Big/ \sum_{i=1}^{n} f(i), \qquad (3.12)$$

где n = *ROULNUM*,

$f(i)$ – *Fitness*-функция i-й хромосомы.

Этот процесс повторяется до тех пор, пока не будет создана популяция, равная 10% от размера всей популяции. При таком отборе члены популяции с более высокой приспособленностью с большей вероятностью будут чаще выбираться, чем особи с низкой приспособленностью.

Этот метод – один из классических, но на практике используется не так часто. Во-первых, такой сэмплинг (выборка) достаточно дорог вычислительно, а во-вторых, этому методу зачастую не хватает разнообразия в получающихся особях. Вторую проблему можно решить, используя ранговый метод, в котором особи сначала сортируются по приспособленности, а затем используется такой же сэмплинг, но вероятность быть выбранной у особи прямо пропорциональна не абсолютному значению ее приспособленности, а её рангу. Даже если особь далеко отрывается от своих конкурентов по значению *Fitness*-функции, ранговый метод всего лишь поставит её на первое место, девальвируя разницу в абсолютных значениях.

Пропорциональная селекция. Фактически это более строгий вариант "рулетки". В последней связь числа вхождений конкретной особи во множество отобранных для репродукции особей с приспособленностью этой особи реализовалась через вероятность отбора $P_{sel}(i)$. В пропорциональном отборе вероятностное звено исключается. Здесь число вхождений особи непосредственно пропорционально приспособленности особи, то есть значению *Fitness*-функции $f(i)$. Таким образом, пропорциональный отбор заключается в отборе 10% первых хромосом от числа популяции из проранжированного списка хромосом по значению *Fitness*-функции.

Операторы скрещивания. В работе реализовано три оператора кроссинговера, каждый из которых выбирается в зависимости от значения случайной величины и выполняется на множестве хромосом-родителей (*pool*), отобранных на этапе селекции:

– вероятностный смешивающий кроссинговер *Mixed Crossover;*
– одноточечный кроссинговер *OnePointCrossover*;
– арифметический кроссинговер *ArithmeticalCrossover*.

Gen Mixed Crossover. Алгоритм генного смешивающего кроссинговера заключается в скрещивании родительских особей на уровне ДНК посредством обмена генами. Вначале вычисляется $\rho=F_1/F_2$, где F_1, F_2 – *Fitness*-функции первого и второго родителей соответственно. Последовательно рассматриваются пары молекул ДНК двух родителей. На уровне ДНК для каждого гена генерируется случайная величина *rnd* со значением в интервале [0,1]. Затем проверяется возможность обмена хромосом соответствующими генами по условию: если $rnd<0.5*\rho$, то обмен генов происходит, иначе нет.

Если первая хромосома оказалась длиннее второй, то оставшаяся часть ДНК, не участвовавшая в скрещивании, будет добавлена к первому потомку. Если вторая хромосома оказалась длиннее первой, то оставшаяся часть ДНК, не участвовавшая в скрещивании, будет добавлена ко второму потомку. Пары родителей из родительского пула для скрещивания выбираются случайным образом.

OnePointCrossover. Этот алгоритм выполняется по классической схеме. Точки кроссинговера в родительских хромосомах выбираются случайным образом.

ArithmeticalCrossover. Данный оператор скрещивания выполняется для всех пар родительского пула. Пары формируются случайным образом. Обмен генами осуществляется посредством вычисления значений генов ДНК потомков на основе значений генов ДНК родителей по формуле:

$$h_{1k} = w*c_{1k}+(1-w)c_{2k}, \quad (3.13)$$

$$h_{2k} = w*c_{2k}+(1-w)c_{1k}, \quad (3.14)$$

где k – номер гена в ДНК;

c_1, c_2 – первая и вторая родительские хромосомы;

h_1, h_2 – первый и второй потомки;

w – коэффициент, значения которого лежат в интервале [0;1], по умолчанию принимается равным 0.5.

Для всех кроссоверов принята единая стратегия выбора родителей из родительского пула, заключающаяся в следующем:

– следование принципу «Право сильнейшего»: хромосома с лучшей пригодностью скрещивается со всеми;

– остальные хромосомы для скрещивания выбираются случайным образом.

Операторы мутации. Размер мутационного пула задается в конфигурационном файле параметром MUTPOOL (нормализованная величина мутантов по отношению к родительскому пулу). Этот пул разбивается на три равные части с целью совершить в каждой части свой вид мутации.

Мутации, как правило, подвергаются хромосомы с лучшими или худшими значениями *Fitness*-функции. Мутация происходит на уровне генов. В работе реализованы три вида оператора мутации, которые выполняются в отдельных частях пула мутантов:

– β-мутация;

– α-мутация;

– мутация действий преобразователя.

α- *и* β-*мутации*. Случайным образом выбирается молекула ДНК в родительской хромосоме. Изменяется ген, ответственный за α-признак или *β*-признак в зависимости от вида мутации. Значения гена выбираются из области допустимых значений.

Мутация действия. Также случайным образом выбирается молекула ДНК в родительской хромосоме. Изменяется ген, ответственный за признак действия в выбранной ДНК, на одно из допустимых значений.

Для каждого поколения вычисляются среднее и максимальное значения fitness-функции популяции. Вычислительные эксперименты показали, что скорость сходимости примерно одинакова как для популяций с невысоким максимальным и невысоким средним значениями *fitness*-функции популяции,

так и с высоким максимальным и высоким средним значениями, с высоким максимальным и невысоким средним значениями *fitness*-функции.

Лучшие хромосомы сохраняются и создают основу родительского пула.

В конце формируется код преобразователя, способного работать в автономном режиме.

3.3.4 Модель сопряжения *IT*

Модель сопряжения со средой UniMod определяется тройкой:

$$IT = < Int, Prototype, Automat >, \quad (3.15)$$

где *Int* – интерпретатор, обеспечивающий перевод внутреннего представления автомата (хромосомы) в модель представления автомата в нотации UniMod. Интерпретатор выполняет несколько проходов анализа хромосомы, в процессе которых создает необходимые объекты (состояния, действия в состояниях, переходы и условия переходов). Данный компонент включает в себя также метод запуска автомата, который соответствует интерпретированной модели. Для прогона особей популяции в инструментальной системе UniMod в её исходный код добавлен метод синхронизации, позволяющий осуществлять их последовательный запуск.

Настройка объектов управления и источников событий реализуется посредством двух компонентов *Prototype* и *Automat*.

Prototype – компонент, отвечающий за создание объекта управления и источника событий преобразователя. Этот компонент содержит прототипы объекта управления и источника событий преобразователя. В его состав включены абстратные методы, обеспечивающие унификацию процесса получения информации из класса управляющего объекта преобразователя и унификацию изменений информации в классе источника событий преобразователя. Компонент содержит классы, обязательные для обеспечения функционирования преобразователя. При появлении новых прикладных задач, решение которых должно реализовываться с помощью генерируемого преобразователя, реализация решения должна осуществляться на базе данных абстрактных классов.

Automat – компонент, обеспечивающий функциональность автомата. Он содержит реализацию управляющего объекта и источника событий преобразователя. При реализации классов используются стандартные и пользовательские библиотеки действий STANDARTLIB и USERLIB соответственно. В библиотеках STANDARTLIB содержатся методы, общие для всех прикладных задач, решаемых созданным преобразователем. Для каждой прикладной задачи создается своя библиотека USERLIB, которая содержит специфичные для неё методы.

3.4 Выводы по главе

Эксперт обычно мыслит некоторыми конструктами, лежащими в основе правил, применяемых для решения некоторой прикладной задачи. В данной главе найден способ представления и формализации конструктивных знаний эксперта. Как результат этого процесса была построена спецификация метода посредством языка XML.

Для генерации систем продукций с использованием построенных спецификаций методов использован генетический алгоритм. Генерация модели преобразователя продукционных правил также производится с использованием генетического алгоритма. Способ представления хромосом популяции в виде деревьев является единым для обоих алгоритмов, хотя структура молекул ДНК хромосом разная. Общим является и то, что в генетическом алгоритме применены однотипные генетические операторы. Подход к их подбору был одинаков и заключался в применении комбинации операторов, а не в использовании какого-то одного оператора. Так, например, на основе вычислительных экспериментов определены комбинации генетических операторов, показывающие лучшие результаты при следующих соотношениях:

1) вероятностный смешивающий кроссинговер – 33%, одноточечный кроссинговер – 33%; арифметический кроссинговер – 34% для скрещивания хромосом;

2) пропорциональная селекция – 80%, турнирная селекция – 20% для отбора хромосом в родительский пул;

3) α-мутация – 40%, β-мутация – 40% и A-мутация (Action-мутация) – 20% для мутации генов внутри хромосомы.

Следует также отметить то, что применение технологий генетического и автоматного программирования позволяет значительно ускорить процесс разработки программного обеспечения, так как в этом случае основной упор в программировании делается только на формирование библиотек классов, содержащих методы реализации различных действий автомата.

4. Активация систем продукций

В любую продукционную систему, кроме модуля правил, в функции которого входит размещение и хранение правил, входит также модуль управления. Главными параметрами, определяющими схему и функциональные характеристики конкретной версии модуля управления, являются: число правил, выбор механизма активации, способ реализации ассоциативного выбора правил, процедура разрешения конфликта и внутреннее представление правил.

Основные составляющие этого модуля – аппарат активации, база правил (П-база) и интерпретатор. В данной работе в качестве интерпретатора правил используется машина логического вывода.

4.1 Механизм активации систем продукций

В системах, не включающих аппарата управления активацией, все правила находятся в активированном состоянии в течение всего процесса. Это означает, что условия применимости каждого правила проверяются при каждом изменении состояния базы. Такая организация естественна при небольшом числе правил, отсутствии достаточных оснований для содержательного структурирования правил по их функциям и удовлетворительной эффективности системы [103].

В общем случае возникает необходимость ограничивать совокупность правил, проверяемых на текущем этапе процесса, для чего в систему вводится аппарат управления активацией.

Очевидно, что существует множество способов организовать такое ограничение. Механизм активации может быть:

– статическим, то есть определенным заранее и не меняющимся в процессе работы;

– динамическим, то есть управляемым ходом процесса;

– смешанным, то есть комбинирующим статические и динамические элементы управления активацией.

Порядок активации может определяться с помощью явно задаваемой структуры, разделяющей множество правил на группы. При этом возможен выбор разных вариантов внутри- и межгрупповой организации правил.

Другим средством управления активацией является включение в правило указания о переходе к одному или нескольким последующим правилам. В случае если такое указание зависит от условия, проверяемого оператором, оно относится к динамической составляющей аппарата активации.

Одним из широко распространенных средств активации правил является использование метаправил (правил над правилами). При таком подходе правила разбиваются на классы, и в зависимости от состояния базы данных активируется тот или иной класс [103].

Механизм активации правил может осуществляться с помощью проверки сопоставленных им специальных условий, выполнение которых

зависит не от текущего состояния базы данных, а от значений особых переменных, посредством которых процесс управляет активацией.

В работе избран смешанный механизм активации, с превалированием статистического над динамическим. Все продукционные правила релевантны некоторому методу и составляют систему продукций метода. В рамках метода они разделены на подсистемы, так как каждый метод может состоять из способов, алгоритмов, то есть система продукций метода, как правило, представляет собой иерархическую систему подсистем продукций. Таким образом, база продукционных правил (П-база) представляет собой вариант внутри- и межгрупповой организации правил. Поэтому порядок активации определяется посредством явно заданной структуры системы продукций.

В работе предлагается схема модуля управления, показанная на рисунке 4.1.

Рисунок 4.1 – Принципиальная схема модуля управления

Формирование входной ситуации и параметров метода. Естественно-языковая обработка научного текста начинается с лексического анализа, который в силу своей тривиальности выполняется обычным алгоритмом. Первый метод, который представлен в виде систем продукций, – это морфологический анализ. Каждый метод может иметь множество входных ситуаций, каждая из которых представляется множеством дизъюнктов. Так, например, при морфологическом анализе входная ситуация описывает текущую лексему. Параметры метода включают наименование метода

(аббревиатура метода), сфера применения *K* (например, *MA* – морфологический анализ), данные для проверки условия применимости продукции.

Выборка системы продукций. Выборка системы продукции происходит по наименованию метода и сфере применимости. Имя продукции состоит из аббревиатуры метода, номера подсистемы и номера продукции в подсистеме.

Выборка дизъюнкта для доказательства. Входная ситуация d_0 задается в виде множества дизъюнктов. Если мощность $|d_0|>1$, то последовательно доказываются все дизъюнкты.

Линейный резолютивный вывод. Машина логического вывода построена в виде системы линейного резолютивного вывода [54,85]. В систему вводится входной дизъюнкт, который доказывается на активной системе продукций и базе фактов. Базы фактов, необходимые для работы текущего метода, выбираются одновременно с системой продукций. Если для всех входных дизъюнктов выведен пустой дизъюнкт, то гипотеза, заложенная в продукционном правиле, считается доказанной.

Поиск и выполнение метода, указанного в постдействии продукции. При успешном доказательстве одной из входных ситуаций выполняется вычислительная процедура, обозначенная идентификатором *H*, который указан в структуре продукционного правила. Все процедуры реализованы в виде методов, записанных в библиотеке методов постдействий. Поиск необходимого метода осуществляется по идентификатору *H*, указанному в продукции.

Разрешение конфликтного множества результатов. В данной работе не определяется конфликтный набор правил. Входная ситуация доказывается на всех правилах. Это означает, что срабатывают процедуры постдействия всех продукций, для которых при доказательстве получен пустой дизъюнкт. Если для одной ситуации получено несколько пустых дизъюнктов, то формируется конфликтное множество результатов работы метода. Для разрешения этого конфликтного множества предусмотрены либо специальные классические системы продукций, либо нечеткие. В первом случае формируется новое множество входных ситуаций, которые доказываются на специальной системе продукций метода, предназначенной для разрешения конфликтного множества результатов. Во втором случае управление передается системе нечеткого логического вывода.

В связи с тем, что производительность работы является одной из главных проблем продукционных систем, то система нечеткого логического вывода работает с небольшим набором правил, а система резолютивного вывода использует механизм распараллеливания процесса доказательства.

Нечеткий логический вывод. Система нечеткого логического вывода построена на основе применения методов нечеткого регулирования [4,43]. Подробное описание системы дано в подразделе 4.4.

Весь описанный процесс повторяется до тех пор, пока не будет закончена естественно-языковая обработка данным методом. Например, при

выполнении метода морфологического анализа завершение наступает после обработки всех лексем текста.

4.2 Автоматная модель аппарата активизации систем продукций

Применим технологию автоматного программирования [101,102,121] для построения модуля управления аппарата активации.

4.2.1 Спецификация автомата

4.2.1.1 Внутреннее представление ядра продукционных правил

Структура продукционного правила описана в разделе 2.1. Внутренняя структура ядра правила зависит от метода логического вывода, применяемого в системе продукций. Так как в работе используется линейный резолютивный вывод, то входная ситуация, ядро продукции метода и утверждений базы фактов должны быть представлены в виде множества дизъюнктов.

Начальное состояние системы продукций должно содержать конъюнкцию терминальных фактов $\bigwedge_{i=1}^{n} P_i^0(e_1,...,e_{m_i})$, которое будем называть входной ситуацией и обозначим d_0. Для доказательства истинности ядра продукции для текущей ситуации d_j будем использовать модифицированный метод линейной резолюции Лавленда, Ковальского и Кюнера. Чтобы применить данный метод, необходимо сформировать множество дизъюнктов Г. Во множество Г включаются дизъюнкты Γ_1, полученные в результате скулемизации ядра продукции, в котором в дизъюнктах могут присутствовать скулемовские функции $g(x_r,...,x_s)$, позволяющие исключить (элиминировать) кванторы существования

$$\Gamma_1=\left\{ \bigvee_{l=1}^{k} P_l\ (x_1,...,x_{i,}g(x_r,...,x_s),x_j..,x_{h_l}) \mid r\geq 1, s\leq i \right\}. \qquad (4.1)$$

При выполнении линейной резолюции входные дизъюнкты C_0 выбираются из множества фактов, задающих входную ситуацию d_0. Конъюнкция фактов $\bigwedge_{i=1}^{n} P_i^0(e_1,...,e_{m_i})$, задающая ситуацию d_0, может быть представлена как множество фактов:

$$d_0=\left\{P_i^0(e_1,...,e_{m_i}) \mid i-количество \quad фактов\right\}. \qquad (4.2)$$

Тогда доказательство истинности ядра продукции $A\Rightarrow B$ на d_0 необходимо производить на множестве дизъюнктов:

$$\Gamma=\Gamma'\cup\bigvee_{i=1}^{n}\overline{P_i^0(e_1,...,e_{m_i})}. \qquad (4.3)$$

Множество дизъюнктов Γ′ есть объединение множества Γ_1 и множеств дизъюнктов, определяющих базы фактов Γ_2, …, Γ_m, релевантных текущему методу обработки научного текста:

$$\Gamma' = \Gamma_1 \cup \Gamma_2 \cup \ldots \cup \Gamma_m. \quad (4.4)$$

Таким образом, $C_0 \in d_0$ является первым центральным дизъюнктом при построении дерева вывода по ядру продукции $(A \Rightarrow B)_i$ некоторой i-й продукции для выбранного факта текущей ситуации d_0. Боковые дизъюнкты B выбираются из множества Γ′. Условием выбора является наличие в дизъюнкте $P \in \Gamma'$ литеры, контрарной самой левой литере центрального дизъюнкта C. Затем дизъюнкты C и B должны быть унифицированы. Для этого используется модифицированный алгоритм унификации. Два дизъюнкта C, B унифицируемы, если для них существует унификатор. Полностью линейный резолютивный вывод, включая алгоритм унификации, описан в разделе 4.3.

Таким образом, во входной алфавит автомата должны войти входная ситуация, состоящая из множества входных дизъюнктов, базы фактов, представленные в виде множества дизъюнктов, системы продукций, ядра которых также представлены в виде множеств дизъюнктов, параметры условия применимости продукций и параметры метода.

4.2.1.2 Описание данных автомата

Разрабатываемый автомат должен настраиваться на выполнение заданного метода. К параметрам, идентифицирующим метод, относятся аббревиатура метода и сфера применения системы продукций. По этим данным осуществляется поиск требуемой системы продукций, так как имя продукции начинается с аббревиатуры метода. Если система продукций имеет иерархическую структуру, то сфера применения определяет элемент этой структуры. В том случае, когда система продукций не имеет иерархической структуры, сфера применения совпадает с аббревиатурой метода.

К параметрам метода относится также список баз фактов, необходимых для его реализации. Любой словарь (база фактов) представляет собой множество трехместных предикатов. В каждом предикате второй аргумент содержит наименование словаря, по которому всегда можно отобрать те словари, которые заданы в списке, при условии, что в списке баз фактов даны наименования словарей, соответствующие вторым аргументам предикатов. Так, например, для морфологического анализа необходимы следующие базы фактов: словарь основ, словарь окончаний, словарь готовых словоформ (СГС), словарь флективных классов (СФК), а также словарь совместимости основы флективного класса и вектора статической морфологической информации (Словарь СМИ), словарь совместимости основы флективного класса и вектора динамической морфологической информации (Словарь ДМИ). В базе фактов «Словарь основ» содержатся предикаты типа: `PAggr(Whole, "Словарь основ", Basis("сущ", "абажур"))`, в базе фактов

«словарь готовых словоформ (СГС)» – *PAggr*(Whole, "СГС", *RWord*("сущ","авизо")). Из примеров видно, что действительно второй аргумент предиката содержит наименование базы фактов.

Для отбора правил из выбранной системы продукций необходимо задать условие применимости *C* продукции. Условие применимости *C* задается в каждом правиле; если оно отсутствует, то параметр *C=null*. В таблице 4.1 приведены примеры условий применимости для разных методов.

Таблица 4.1 – Примеры условия применимости

Метод	N правила	Левый параметр			Операция	Правый параметр		
		Переменная	Тип данного	Значение		Переменная	Тип данного	Значение
MA	1	*e*	String	в файле	=	*null*	null	" "
…	…	…	…	…	…	…	…	…
CO	7	PrT_i	set	в файле	⊃	*PrN*	set	в файле
…	…	…	…	…	…	…	…	…
CO	17	*x*	int	в файле	>	*y*	int	в файле
SRWP	3	*k*	String	" "	≠	*null*	null	*null*
SRWP	3	*q*	set	в файле	≠	*null*	null	∅
…	…	…	…	…	…	…	…	…
SRWP	11	*k*	String	в файле	≠	*null*	null	" "
…	…	…	…	…	…	…	…	…
SRWP	23	*q*	set	в файле	≠	*null*	null	∅
…	…	…	…	…	…	…	…	…

В первой строке таблицы 4.1 приведен пример для продукции морфологического анализа (*MA*), определяющей ситуацию, в которой анализируемая лексема является готовой словоформой, условие применимости имеет вид: окончание *e* = " ". Для метода «Соединение онтологий» (*CO*) пример условия применимости правила имеет вид: $PrT \supset PrN$, в котором проверяется включение множества свойств номенклатуры во множество свойств терминосистемы. Условие применимости 3-го правила метода распознавания семантического отношения «Целое-часть» (SRWP)

включает проверку непустой строки композиционного словосочетания ($k \neq$ “ ”) и непустого списка элементов ($q \neq \varnothing$).

Таким образом, задание параметров условия применимости заключается в задании значений левого и правого параметров, причем у одного правила может иметься составное условие. Из этого следует, что при задании параметров условия применимости в зависимости от описываемой ситуации возможны варианты:

– задаются значения левого и правого параметров;

– задается значение только одного левого параметра, а значение правого параметра будет “null”;

– задаются значения левого и правого параметров более одного раза;

– задается значение левого параметра более одного раза.

Задание входной ситуации тоже рассмотрим на примерах. Для морфологического анализа входная ситуация может иметь пять вариантов гипотез:

1) возможно, лексема является готовой словоформой:

{*PAggr*(“Whole”, *p*, “автоматная”),
PAggr(“Part”, “автоматная”, d_1),
PAggr(“Whole”, ″СГС″, *RWord*(c_1,″автоматная″)};

2) возможно, лексема состоит только из основы:

{*PAggr*(“Whole”, *p*, “автоматная”),
PAggr(“Whole”, ″Словарь основ″, *Basis*(c_1, ″автоматная″)),
PAggr(“Whole”, ″Словарь окончаний″, *End*(c_3, ′ ′))};

3) возможно, последняя буква является окончанием, а остальная часть лексемы является основой:

{*PAggr*(“Whole”, *p*, “автоматная”),
PAggr(“Whole”, ″Словарь основ″, *Basis*(c_1, ″автоматна″)),
PAggr(“Whole”, ″Словарь окончаний″, *End*(c_3, ′я′))};

4) возможно, две последние буквы являются окончанием, а остальная часть лексемы является основой:

{*PAggr*(“Whole”, *p*, “автоматная”),
PAggr(“Whole”, ″Словарь основ″, *Basis*(c_1, ″автоматн″)),
PAggr(“Whole”, ″Словарь окончаний″, *End*(c_2, ′ая′))};

5) возможно, три последние буквы являются окончанием, а остальная часть лексемы является основой. Одна из этих гипотез должна быть истинной:

{*PAggr*(“Whole”, *p*, “автоматная”),
PAggr(“Whole”, ″Словарь основ″, *Basis*(c_1, ″автомат″)),
PAggr(“Whole”, ″Словарь окончаний″, *End*(c_3, ′ная′))}.

Входная ситуация для метода извлечения знаний из терминологического словаря о семантическом отношении «Целое-часть» представляется одним множеством:

```
{PHier(“Value”,k1,“Составной частью архитектуры ЭВМ”),
PHier(“Value”,v1,“является”),
PHier(“Value”,s3,“структура её памяти”),
```

```
PHier("Value",s1,"Составной частью"),
PHier("Value",s2,"архитектуры ЭВМ"),
PHier("Value",tx1,"частью"),
PHier("Value",c1, "Тв"),
PHier("Value",c2, "Род"),
PHier("Value",i1,1)}
```

Данное множество состоит только из предикатов, в котором переменная (второй аргумент) получает значение (третий аргумент). Большая часть методов имеет именно такой вид входной ситуации.

В качестве средства описания исходной информации о методе естественно-языковой обработки снова используем язык XML. Для рассматриваемого автомата используются два файла "CommonInfMethods.xml" и "InpMethodMAInf.xml". Первый файл единственный, в нем описываются общие данные о методах. Файл второго типа создается для каждого метода и содержит информацию, необходимую для выполнения метода.

Пример файла "CommonInfMethods.xml"

```
<?xml version="1.0" encoding="utf-8"?>
<CommonInformationMethod>
<order method>
  <method numder = 1>
    <name short_name="MA" useArea="MA" />
    <data IdentFile="InpMethMAInf.xml"/>
    <object analysis name="лексема"/>
    <MethodSolvationSetConflict IdentFile="SolveSetConflMA"/>
    <list dictionary>
      <dictionary number="1" name="СГС"/>
      <dictionary number="2" name="Словарь основ"/>
      <dictionary number="3" name="Словарь окончаний"/>
      <dictionary number="4" name="СФК"/>
      <dictionary number="5" name="Словарь совместимостей"/>
    </list dictionary>
  </method>
  <method numder = 2>
    <name short_name="StatA" useArea="SelPhrase" />
    <data IdentFile="InpMethStatAInf.xml"/>
    <object analysis name="лексема"/>
    <object analysis name="словосочетание"/>
    <MethodSolvationSetConflict IdentFile="SolveSetConflStatA"/>
  </method>
  <method numder = 3>
  …
  </method>
  …
</order method>
<scientifical text name="???"/>
</CommonInformationMethod>
```

Пример файла "*InpMethodMAInf.xml*".

```
<?xml version="1.0" encoding="utf-8"?>
<ObjectAnalysis name="лексема" number=1 value="автоматная"/>
<!-- параметры условий применимости для лексемы "автоматная" -->
  <rule production>
    <rule number="1">
      <params condition number="1">
        <left type=variable value="10">
        <operation compare value="=">
        <right type=variable value="10">
```

```
        </params condition>
        <operation connection value="AND">>
        <params condition number="2">
          <left type=set value="{a,b,c,d}">
          <operation compare value="≠">
          <right type= set value="∅">
        </params condition>
        …
      </rule>
      <rule number="2">
        <params condition number="1">
          <left type=set value="{1,2,3,4,5}">
          <operation value="∩">
          <right type=set value="{2,4,5,7}">
          <operation compare value="≠">
          <right type=set value="∅">
        </params condition>
      </rule>
      …
    </rule production>
    <!—множество входных ситуаций -->
    <InputSituation>
      <disjunctset value="first">
      {PAggr("Whole", p, "автоматная"),
        PAggr("Part", "автоматная", d1),
        PAggr("Whole", "СГС", RWord(c1,"автоматная")}
    </disjunctset>
    <disjunctset value="second">
      {PAggr("Whole", p, "автоматная"),
        PAggr("Whole","Словарь основ",Basis(c1,"автоматная")),
        PAggr("Whole", "Словарь окончаний", End(c3, ' '))}
      </disjunctset>
      <disjunctset value="third">
      {PAggr("Whole", p, "автоматная"),
        PAggr("Whole","Словарь основ",Basis(c1,"автоматна")),
        PAggr("Whole", "Словарь окончаний", End(c3, 'я'))}
      </disjunctset>
      <disjunctset value="four">
        {PAggr("Whole",p,"автоматная"),
        PAggr("Whole","Словарь основ",Basis(c1,"автоматн")),
        PAggr("Whole","словарь окончаний",End(c3,"ая"))}
      </disjunctset>
      <disjunctset value="fifth">
        {PAggr("Whole",p,"автоматная"),
        PAggr("Whole","Словарь основ",Basis(c1,"автомат")),
        PAggr("Whole","словарь окончаний",End(c3,"ная"))}
      </disjunctset>
    </InputSituation>
  </ObjectAnalysis>
```

Такой файл для морфологического анализа должен содержать информацию (параметры условия применимости и множество входных ситуаций, в данном случае – по пять на каждую лексему) обо всех лексемах научного текста.

4.2.2 Модель автомата модуля управления

В соответствии с принципиальной схемой модуля управления на рисунке 4.2 показана его автоматная модель. Конечный автомат *FA*, созданный в нотации Unimod, представляет собой кортеж вида [101]:

$$FA=<S, E, G, Z, S_0, S_f, F>, \quad (4.5)$$

где S – множество состояний;

E – множество событий;

G – множество условий перехода;

Z – множество выходных воздействий;

S_0 – начальное состояние;

S_f – заключительное состояние.

$F(s,e,g,a)$ – функция перехода, зависящая от текущего состояния, события, условия перехода и выходного воздействия. Это означает, что переход из текущего состояния s в новое срабатывает тогда, когда было сгенерировано событие e, и значение переменной объекта управления совпало с указанным значением в условии перехода g, сопровождающееся выходным воздействием z. Программа, написанная с помощью Unimod и соответствующая концепции автоматного программирования, содержит схему связей, состоящую из источника событий, системы управления и объектов управления, как и в SWITCH-технологии.

Источник событий информирует систему управления о завершении выполнения действий в состоянии. К выходным воздействиям автомата относятся выходные переменные и переменные, необходимые для вычисления условия перехода, на основе которых формируется значение переменной события. Далее производится проверка переменной события, и осуществляется соответствующий переход. Таким образом, объект управления инкапсулирует описание выходных воздействий и условий перехода.

Модель автомата (рис. 4.2) состоит из пяти состояний и пяти вложенных автоматов, один из которых относится к третьему уровню вложенности:

1) *SetActiveProdSystem* – автомат, предназначенный для выбора правила продукции и баз фактов (формирование множества Γ_1 и Γ_2, …, Γ_k, где k – количество баз фактов, используемых методом для логического вывода);

2) *SelectInSituattionSet* – автомат, предназначенный для формирования и выгрузки в рабочую память множества дизъюнктов d_0 входных ситуаций;

3) *SelDisjunctAndResolution* – автомат, обеспечивающий логический вывод для текущего правила на всех дизъюнктах входной ситуации и всех входных ситуациях;

4) *SolvationOfConflictSet* – автомат, предназначенный для определения системы разрешения конфликтного множества, передачи ей управления и подготовки необходимой информации;

5) *FuzzyResolution* – автомат, предназначенный для вызова системы нечеткого логического вывода.

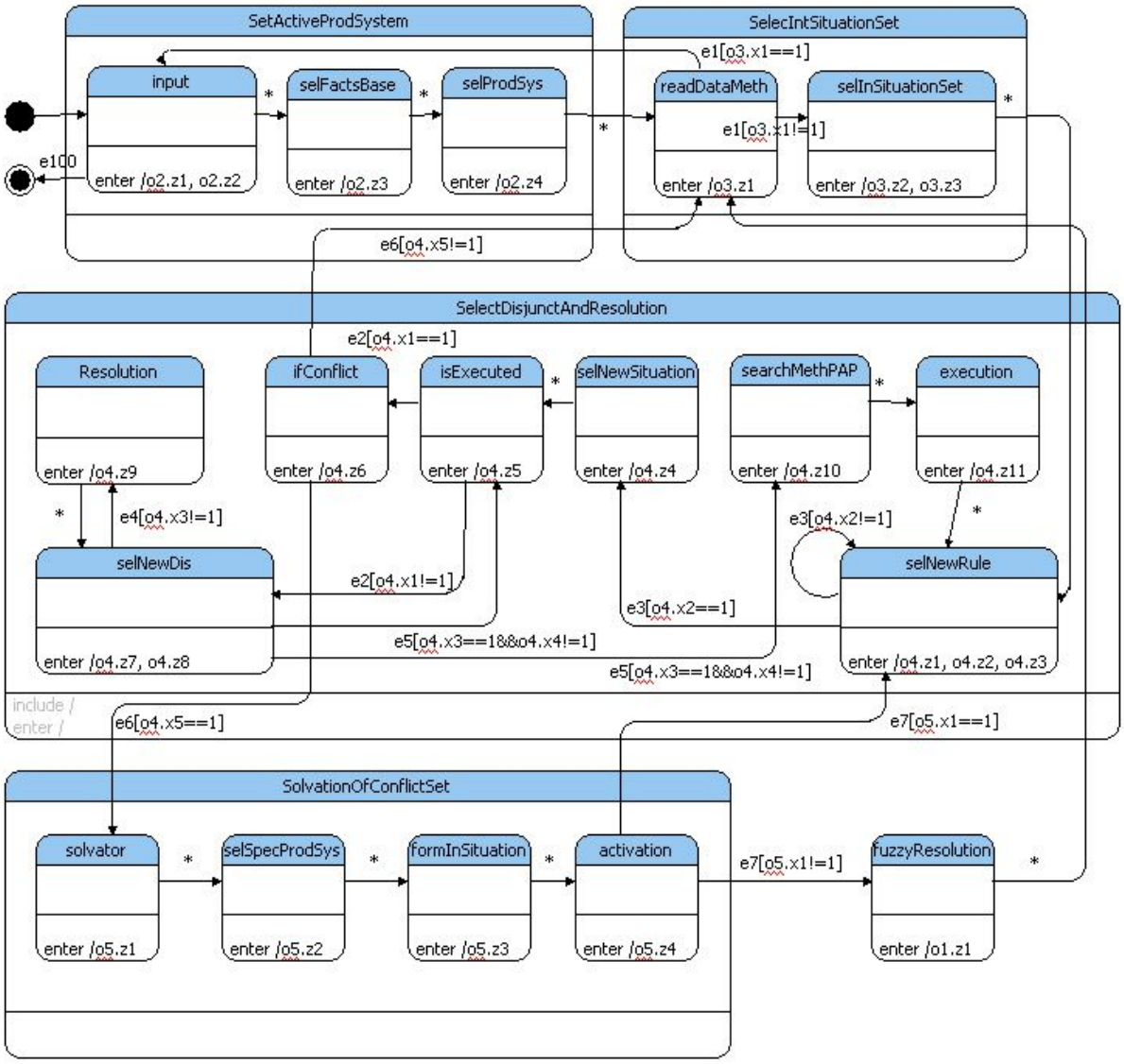


Рисунок 4.2 – Модель автомата модуля управления

В автомате почти в половине случаев осуществляется безусловный переход между состояниями, остальные переходы осуществляются по событиям, приведенным в таблице 4.2.

Автомат *SetActiveProdSystem*. Автомат имеет три состояния. В состоянии «Input» выполняются следующие выходные воздействия:

o2.z1 – ввод информации в буфер о порядке выполняемых методов, объектах их анализа и идентификаторах файлов, содержащих информацию о методе, включая информацию об объекте анализа метода из файла *CommonInfMethods.xml*;

Таблица 4.2 – Основные события автомата *ActivationMachine*

Событие	Пояснение
e1[o3.x1!=1]	не достигнут конец файла *InpMeth#Inf.xml*
e1[o3.x1==1]	достигнут конец файла *InpMeth#Inf.xml*
e2[o4.x1!=1]	резолюция выполнена не для всех входных ситуаций
e2[o4.x1==1]	резолюция выполнена для всех входных ситуаций
e3[o4.x2!=1]	условие применимости правила ложно
e3[o4.x2==1]	условие применимости правила истинно
e4[o4.x3!=1]	резолюция выполнена не для всех дизъюнктов текущей входной ситуации
e4[o4.x3==1]	резолюция выполнена для всех дизъюнктов текущей входной ситуации
e5[o4.x4!=1]	не для всех дизъюнктов текущей входной ситуации получен пустой дизъюнкт
e5[o4.x4==1]	для всех дизъюнктов текущей входной ситуации получен пустой дизъюнкт
e6[o4.x5!=1]	конфликтное множество пустое, получен корректный результат
e6[o4.x5==1]	конфликтное множество не пустое
e7[o5.x1!=1]	средством разрешения конфликтного множества является классическая система продукций
e7[o5.x1==1]	средством разрешения конфликтного множества является нечеткая система продукций

o2.z2 – определение текущего метода, подлежащего выполнению, поиск идентификатора файла *InpMeth#Inf.xml*, содержащего информацию о методе, чтение информации о методе и запись её в новый буфер в виде структуры. Если выполнены все методы, то переход по событию e100 на конечное состояние автомата *ActivationMachine*, в противном случае переход в состояние *selFactsBase*.

В состоянии *selFactsBase* в соответствии со списком баз фактов, содержащимся в слоте *ListDictionary* файла *CommonInfMethods.xml*, выполняется выходное воздействие o2.z3, заключающееся в выборе и записи в рабочую память фактов словарей, формируя, таким образом, множество дизъюнктов $\Gamma_2, \ldots, \Gamma_k$.

В состоянии *selProdSys* выполняется выходное воздействие o2.z4 – выбирается активная система продукций по аббревиатуре метода. Аббревиатура задается в атрибуте *short_name* тега *name* файла *CommonInfMethods.xml*. Подсистема продукций уточняется по сфере применения, которая задается в том же теге в атрибуте *useArea*. Таким образом, формируется и дописывается в рабочую память множество дизъюнктов Γ_1.

Автомат *SelectInSituationSet*. Автомат включает два состояния. В первом состоянии «*readDataMeth*» выполняется выходное воздействие o3.z1 – считывается информация об объекте анализа выполняемого метода из XML-файла *InpMeth#Inf.xml*. При достижении конца файла генерируется событие e1[o3.x1==1], и осуществляется переход в состояние *Input* автомата *SetActiveProdSystem*; если конец файла не достигнут, то генерируется

событие e1[o3.x1!=1], и выполняется переход в состояние *selInSituationSet*. Этому состоянию соответствуют следующие действия:

o3.z2 – осуществляется выбор множества входных ситуаций d_0, соответствующих текущему значению объекта анализа. Например, в случае морфологического анализа объектом анализа является лексема. По каждой лексеме в файле записаны четыре возможные ситуации.

o3.z3 – запись множества входных ситуаций об объекте анализа в буфер. За одно обращение к состоянию в рабочую память загружается множество ситуаций только для одной лексемы. Всего в процессе работы должно быть сформировано множество ситуаций с мощностью, равной $4\times n\times m$, где n – количество предложений в научном тексте, m – количество лексем в предложении.

В основном все переходы между состояниями автоматов *SelectInSituationSet* и *SetActiveProdSystem* носят безусловный характер. По окончании работы автомата *SetActiveProdSystem* управление передается в состояние «*readDataMeth*», а автомата *SelectInSituationSet* – в состояние *selNewRule* автомата *SelDisjunctAndResolution*.

Автомат *SelDisjunctAndResolution*. Автомат включает восемь состояний. В состоянии *selNewRule* происходят следующие выходные воздействия:

o4.z1 – выборка нового правила из активной системы продукций;

o4.z2 – чтение файла *InpMeth#Inf.xml*, выборка из тегов *params condition* значений параметров условия применимости *C*, соответствующих текущему объекту анализа;

o4.z3 – вычисление условия применимости *C* правила. Если *C=true*, то генерируется событие e3[o4.x2==1], и осуществляется переход на состояние *selNewSituation*, в противном случае – выполнение выходных воздействий o4.z1 и o4.z2, и снова вычисление условия применимости (петля при генерации события e3[o4.x2!=1]).

В состоянии *selNewSituation* осуществляется действие o4.z4, заключающееся в выборе новой ситуации из множества ситуаций, находящихся в рабочей памяти.

В состоянии *isExecuted* проверяется, все ли возможные ситуации доказаны для текущего правила (выходное воздействие o4.z5). Если это истинно, то управление передается в состояние *ifConflict* (событие e2[o4.x1==1]), если нет (событие e[o4.x1!=1]), то *selNewDis*.

В состоянии *selNewDis* выбирается новый дизъюнкт из текущей ситуации, и если не все дизъюнкты перебраны, то генерируется событие e4[o4.x3!=1], и автомат переходит в состояние *Resolution* (выходное воздействие o4.z6). В этом состоянии осуществляется вызов автомата *DedLogRes* (выходное воздействие o4.z9), который осуществляет линейный резолютивный вывод. Переходы между состояниями *selNewDis* $\leftrightarrow$ *Resolution* выполняются до тех пор, пока не будут доказаны все дизъюнкты текущей ситуации (при генерации события e4[o4.x3!=1]). Если на всех дизъюнктах одной ситуации получен пустой дизъюнкт, то гипотеза, заложенная в

текущем правиле и текущей входной ситуации, доказана. Таким образом, выходное воздействие o4.z7 заключается в подсчете количества *k*, для которых получен пустой дизъюнкт и сравнение *k* с количеством дизъюнктов в текущей входной ситуации. Если они совпали, то генерируется событие e5[o4.x3==1 && o4.x4==1], и осуществляется переход в состояние *searchMethPAP*, иначе генерируется событие e5[o4.x3==1 && o4.x4!=1], и осуществляется переход в состояние *selNewSituation*. Если пустой дизъюнкт получен для нескольких входных ситуаций, то формируется конфликтное множество (выходное воздействие o4.z8).

В состоянии *searchMethPAP* осуществляется поиск метода, указанного в постдействии *H* текущего продукционного правила (выходное воздействие o4.z10). В состоянии *execution* осуществляются вызов и выполнение найденного метода – выходное воздействие o4.z11. Затем осуществляется переход в состояние *selNewRule*.

В состоянии «ifConflict» проверяется конфликтное множество (выходное воздействие o4.z6), если оно непустое, то генерируется событие e6[o4.x5==1], и управление передается автомату *SolvationOfConflictSet*. Если конфликтное множество пустое, то генерируется событие e6[o4.x5!=1], и выполняется переход в состояние *readDataMeth* автомата *ConstructInSituationSet*.

Автомат *SolvationOfConflictSet*. Автомат включает четыре состояния. В состоянии *solvator* происходит определение средства разрешения конфликтного множества – выходное воздействие o5.z1. В качестве средства разрешения может быть либо специальная классическая система продукций, либо нечеткая систем продукций, и, соответственно, должен выполняться либо логический резолютивный вывод (переход в состояние *selNewRule* автомата *SelDisjunctAndResolution*), либо нечеткий логический вывод (переход в состояние *fuzzyLogic* автомата *ActivationMachine*). После определения одного из указанных средств, что отмечается в глобальной переменной *SCS,* осуществляется переход в состояние *selSpecProdSys*. В этом состоянии осуществляется поиск специальной системы продукций (классической или нечеткой), предназначенной для разрешения данного конфликтного множества – выходное воздействие o5.z3. Затем выполняется безусловный переход в состояние *formInSituation*, в котором осуществляется формирование входной ситуации – выходное воздействие o5.z2. И снова осуществляется безусловный переход в состояние *activation*. В нем специальная система продукций и входные ситуации помещаются в рабочую память, и в соответствии со значением переменной *SCS* генерируется событие e7[o5.x1==1], при этом управление передается либо в состояние *selNewSituation* автомата *SelDisjunctAndResolution*, либо генерируется событие e7[o5.x1!=1], затем управление передается в состояние *fuzzyLogic* автомата *ActivationMachine* (выходное воздействие o5.z4).

В состоянии *fuzzyLogic* основного автомата *ActivationMachine* осуществляется вызов системы нечеткого логического вывода *FuzzyRegulator*. Из этого состояния управление передается снова в состояние

readDataMeth автомата *ConstructInSituationSet* на цикл обработки следующего или нового объекта анализа.

В следующем разделе описана модель системы резолютивного вывода, являющейся ядром аппарата активации.

4.2.3 Модель системы резолютивного вывода

Решение задач резолютивного вывода сводится к поиску наилучшего пути решения. При этом сложность поиска возрастает экспоненциально по мере продвижения по дереву вывода. Поэтому при решении задач, характеризующихся экспоненциальным ростом пространства поиска, особое значение приобретает проблема эффективности процедур поиска. Вследствие этого в работе используется эволюционная стратегия [24,73] для генерации наиболее эффективной эвристики для заданной выборки множества дизъюнктов.

Имеется семейство множеств дизъюнктов $\Gamma = \{\Gamma_i | \ i = 1..n, \ n$ – количество методов$\}$. Назовем множество дизъюнктов метода выборкой. Тогда задача состоит в том, чтобы для каждой выборки $\Gamma_i \in \Gamma$ сгенерировать конечный автомат, позволяющий осуществлять резолютивный вывод и обладающий высокой эффективностью именно для конкретной выборки.

4.2.3.1 Основные положения генетического алгоритма

В работе за основу взяты следующие положения:

1. Хромосома (модель автомата) представляется в виде графа переходов.
2. Вершины графа бывают двух видов *X* и *Y*. Множество вершин *X* предназначено для выбора центрального дизъюнкта и содержит обозначения множеств начальных и конечных состояний; множество *Y* используется для сортировки множества центральных дизъюнктов.
3. Множество вершин *X* назовем молекулами *X*-ДНК, а множество вершин *Y* – молекулами *Y*-ДНК. Молекулы *X*-ДНК и *Y*-ДНК должны составлять замкнутую цепочку переходов.
4. Молекулы ДНК не должны повторяться в особи.
5. Молекула ДНК состоит из трех генов.
6. Каждый ген представляется в виде целого числа.
7. Первая популяции $P(0)$ генерируется случайным образом, при этом значение каждого гена выбирается из соответствующей области допустимых значений.
8. Каждое новое поколение $P(t)$ генерируется при помощи оператора кроссинговера. Родительские пары выбираются оператором репродукции по методу заданной шкалы.
9. В каждой новой хромосоме повторяющиеся молекулы, если таковые есть, подвергаются мутации.
10. Все повторяющиеся хромосомы из популяции удаляются.

11. Оценивание хромосом и их ранжирование осуществляются на основе использования внешнего для генетического алгоритма объекта – инструментальной системы UniMod.

Структура хромосомы. Хромосома состоит из динамического количества молекул ДНК каждого типа. Структура молекулы состоит из трех генов (рис. 4.3):

1) ген1 – множитель перехода, интервал значений которого лежит в диапазоне от 1 до 3, используется для указания того, в какие состояния следует переходить чаще при наличии других вариантов.

2) ген2 – новое состояние, интервал значений которого для молекул вида X лежит в диапазоне от 1 до 27, а для молекул типа *Y* – от 1 до 27;

3) ген3 – текущее состояние, интервал значений которого для молекул вида X лежит в диапазоне от 1 до 27, а для молекул типа *Y* – от 1 до 27.

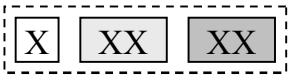


Рисунок 4.3 – Структура молекулы ДНК

На рисунке 4.4 показан абстрактный пример структуры хромосомы.

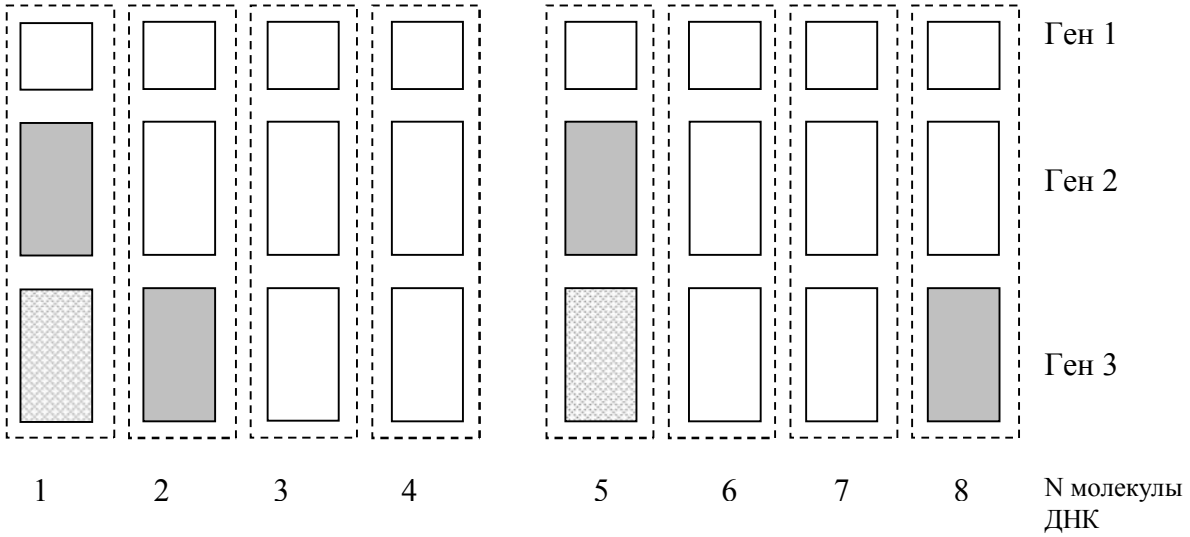


Рисунок 4.4 – Пример структуры хромосомы

Примечание: серым цветом закрашены гены, содержащие начальное и конечные состояния, заштрихованные гены содержат состояния «склейки».

На рисунке 4.4 гены белого цвета образуют *Y*-ДНК, остальные гены – *X*-ДНК.

Первичные правила проверки хромосом на корректность, применяемые при создании новой популяции:

1) во множестве молекул должна присутствовать, по крайней мере, одна хромосома, имеющая значение 2-го гена, равное 0 (начальное состояние);

2) во множестве молекул должна присутствовать, по крайней мере, одна хромосома, имеющая значение 3-го гена, равное 0 (начальное состояние);

3) во множестве молекул должна присутствовать, по крайней мере, одна хромосома, имеющая значение 2-го гена, равное 27 (конечное состояние);

4) во множестве молекул должна присутствовать, по крайней мере, одна хромосома, имеющая значение 2-го гена, равное 27 (конечное состояние).

4.2.3.2 Генетический алгоритм

Создание начальной популяции $P(0)$ осуществляется случайным образом. Проверка выполнения первичных правил на наличие в хромосоме начального и конечного состояний, мутация некорректных молекул осуществляются непосредственно при генерации популяции.

Основные моменты построения графа переходов. Граф переходов представляет собой циклический граф, в котором есть одно начальное состояние и два конечных, идентифицирующих успешное и неуспешное завершение работы. Успешное завершение достигается только в случае нахождения пустого дизъюнкта во множестве дизъюнктов. Неуспешное завершение работы возникает в случаях:

1) истечения заданного количества резольвирований;

2) опустошения входного множества дизъюнктов.

Каждый переход в графе имеет свой коэффициент прохождений и весовой коэффициент. Значения коэффициента прохождений меняются в интервале [1..100] и служат для справедливого прохождения всех вершин графа. Каждый раз, когда осуществляется переход, значение весового коэффициента умножается на значение элемента матрицы переходов и добавляется к соответствующему коэффициенту прохождений. Предпочтение в выборе вершины отдается той вершине, в которой значение коэффициента прохождений больше. Когда значение коэффициента прохождений становится больше 100, значение сбрасывается в 1.

На каждый граф накладываются следующие ограничения:

1) граф не должен содержать состояний, из которых нет переходов;

2) не должно быть пары таких состояний *А*, *Б*, которые замыкаются на самих себя;

3) имеющиеся петли в графе переходами не являются;

4) в графе должен быть переход в 0 состояние;

5) только из 0-го состояния возможны выход из автомата и вход в него;

6) ход вывода должен затрагивать как можно большее количество состояний в графе.

Все состояния можно разделить на состояния, используемые для оценки и ранжирования множества центральных дизъюнктов, и служебные состояния, к которым относятся начальное состояние, множество конечных состояний, состояние, в котором осуществляются унификация и склейка

центрального дизъюнкта с одним из множества боковых. В состоянии склейки выполняются следующие действия:

1) проверка дизъюнктов на унифицируемость;
2) склейка дизъюнктов;
3) обрамление литер;
4) удаление обрамленных литер;
5) проверка на редуцируемость и достижимость пустого дизъюнкта.

На базе вышеуказанных состояний формируется исполняемый в среде UniMod автомат. Среда UniMod используется для проверки достижимости конечного состояния и расчета *Fitness*-функции.

Для осуществления выбора следующей вершины, на которую осуществляется переход, используются две матрицы: матрица прохождений графа и матрица зацикливаний. Матрица зацикливаний содержит целочисленные значения [1,100], равномерно покрывающие все вершины. Элементы матрицы инициализируются значением 1, когда автоматом осуществляется переход в одну из альтернативных вершин, просматриваются значения соответствующих элементов матрицы прохождений на установление вершины, имеющей наибольший коэффициент. После перехода осуществляется инкремент элемента матрицы на значение, соответствующее весу дуги графа. В случае достижения значения матрицы, равного 100, это значение сбрасывается в 1, и инкремируется значение альтернативного элемента перехода.

Матрица зацикливаний служит для останова некорректных автоматов, прошедших через блок селекции. В каждой вершине, не соответствующей служебным состояниям, применяются методы ранжирования дизъюнктов, представляющие собой универсальный алгоритм сортировки. При каждом прохождении вершины в графе переходов происходит продвижение лучшего дизъюнкта к началу списка. При этом оцениваются такие параметры, как:

1) глубина дизъюнкта;
2) количество непройденных боковых дизъюнктов;
3) количество обрамленных литер и другие параметры.

Оценивание. Оценивание состоит из двух этапов: оценки автоматов на выполнимость и оценки приспособленности особей.

Оценка автоматов на выполнимость. Данный вид оценки особей осуществляется с помощью стандартных функций UniMod и заключается в проверке:

1) наличия ошибок в объектном коде;
2) наличия ошибок в автоматах;
3) достижимости конечного состояния.

Для проверки объектного кода применяется класс *DefaultCompilationListener*, объект которого создан в классе *Run*. В конструктор класса подается откомпилированная модель автомата (объект класса *StateMachineCompiler*), после чего вызывается функция *getErrors*(), которая сообщает о наличии ошибок в модели.

Определение наличия ошибок в автоматах осуществляется с помощью функций *validateStructure*() и *getAttainableStates*() класса *StateMachineValidator*, которые верифицируют структуру автомата (наличие противоречивых переходов).

Проверка достижимости конечного состояния осуществляется посредством функции *getName*() класса *StateMachineValidator*, возвращающей имя нового состояния. Если условие *getName*()==“*finish*” не выполнится ни разу, то построенный автомат является некорректным. Некорректные хромосомы мутируются в течение одной итерации; если в дальнейшем хромосома осталась некорректной, то она удаляется из популяции.

Оценка приспособленности особей или расчет *Fitness*-функции осуществляется по формуле:

$$F(a_i) = U(a_i) + 1/k_1(a_i) + 1/k_2(a_i) + 1/k_3(a_i), \quad (4.6)$$

где a_i – текущая особь;

$U(a_i)$ – количество успешных унификаций;

$k_1(a_i)$ – среднее количество литер во множестве С;

$k_2(a_i)$ – среднее количество необрамленных литер во множестве С;

$k_3(a_i)$ – соотношение средней арности термов к количеству константных символов в них.

Особи ранжируются в соответствии со значением $F(a_i)$ по убыванию.

Оператор селекции. Особи для выполнения операторов мутации и кроссинговера выбираются случайным образом с преимуществом выбора хромосом из начала списка.

Оператор мутации. Используется двухточечный оператор мутации. Каждое значение, генерируемое оператором, должно соответствовать правилам формирования молекул.

Оператор мутации не применяется к молекулам, содержащим начальное и конечное состояния и состояние склеивания дизъюнктов. Оператор мутации используется как для модификации молекул, так и для создания новой хромосомы.

Оператор кроссинговера. В алгоритме используется двухточечный оператор кроссинговера. У особи родителей P_1 и P_2, имеющей меньшую длину, случайным образом выбираются две позиции. В потомок P'_1 копируются молекулы ДНК хромосомы P_1 от начала до первой точки кроссинговера, затем молекулы ДНК хромосомы P_2 – от первой точки кроссинговера до конца молекулы, затем молекулы ДНК хромосомы P_1 – от начала части второй молекулы до второй точки кроссинговера, затем молекулы ДНК хромосомы P_2 – от второй точки кроссинговера до конца. Потомок P'_2 формируется в инверсном порядке.

Если в результате операции из хромосомы потомка выпадают гены, содержащие начальное, конечное состояния или состояние склейки, то молекулы одного из родителей, содержащие их, также копируются в эту хромосому. В случае если у потомка оказалось более двух ген, содержащих начальное состояние, одна из молекул, содержащих это состояние, удаляется.

4.2.3.3 Конечный автомат системы резолютивного вывода

Сгенерированный автомат с лучшим значением функции приспособленности становится конечным автоматом Мура, реализующим линейный резолютивный вывод для выборки Γ_i.

Модель системы резолютивного вывода построена в виде вычислительной сети, которая состоит из сервера заданий и вычислительных узлов. Каждый вычислительный узел представляет собой конечный автомат Мура.

Количество вычислительных узлов создается в соответствии с потребностью вычислительных ресурсов для текущей задачи. По умолчанию для одной задачи создается один узел.

Инициализация системы осуществляется с помощью передачи множества входных дизъюнктов серверу заданий. Передача происходит с помощью протокола межсетевого взаимодействия. После завершения логического вывода вычислительный узел передает на сервер результат доказательства: успех, неуспех.

4.2.4 Нечеткий логический вывод

Рассмотрим нечеткий логический вывод на основе использования формализма нечетких продукционных правил [4,43].

4.2.4.1 Нечеткие продукции

Нечеткие правила продукций во многом близки логическим моделям представления знаний. В общем случае под *правилом нечеткой продукции* или просто — *нечеткой продукцией* понимается выражение следующего вида:

$$(i):Q;P;A\Rightarrow B;S,F,N, \qquad (4.7)$$

где

(i) – имя нечеткой продукции;

Q – сфера применения нечеткой продукции;

P – условие применимости ядра нечеткой продукции;

$A\Rightarrow B$ – ядро нечеткой продукции, в котором A – условие ядра (или антецедент), B – заключение ядра (или консеквент), «$\Rightarrow$» – знак логической секвенции (или следования);

S – метод или способ определения количественного значения степени истинности заключения ядра;

F – коэффициент определенности или уверенности нечеткой продукции;

N – постусловия продукции.

По аналогии с обычным правилом продукции в качестве имени (i) нечеткой продукции может выступать та или иная совокупность букв или символов, позволяющая однозначным образом идентифицировать нечеткую продукцию в системе нечеткого вывода или базе нечетких правил. В качестве имени нечеткой продукции может использоваться ее номер в системе.

Сфера применения нечеткой продукции Q, условие применимости ядра нечеткой продукции P и постусловие нечеткой продукции N определяются аналогично классической продукции. Также и ядро $A \Rightarrow B$ является центральным компонентом нечеткой продукции. Ядро продукции записывается в той же форме: «Если A, то B», где A и B – некоторые выражения нечеткой логики, которые наиболее часто представляются в форме нечетких высказываний, при этом секвенция интерпретируется в обычном логическом смысле как знак логического следования заключения B из условия A. В качестве выражений A и B могут использоваться составные логические нечеткие высказывания, то есть элементарные нечеткие высказывания, соединенные нечеткими логические связками, такими, как нечеткое отрицание НЕ, нечеткая конъюнкция И, нечеткая дизъюнкция ИЛИ.

S – метод или способ определения количественного значения степени истинности заключения B на основе известного значения степени истинности условия B. Данный способ в общем случае определяет, так называемую, *схему* или алгоритм нечеткого вывода в продукционных нечетких системах и называется также *методом композиции* или *методом активации*. F – коэффициент определенности или уверенности выражает количественную оценку степени истинности или относительный вес нечеткой продукции. Коэффициент уверенности принимает свое значение из интервала [0,1] и часто называется весовым коэффициентом нечеткого правила продукции.

Продукционная нечеткая система или *система нечетких правил продукций* представляет собой некоторое согласованное множество отдельных нечетких продукций или правил нечетких продукций.

Основная проблема приближенных рассуждений с использованием нечетких правил продукций заключается в том, чтобы на основе некоторых нечетких высказываний с известной степенью истинности, которые являются условиями нечетких правил продукций, оценить степень истинности других нечетких высказываний, являющихся заключениями соответствующих нечетких правил продукций.

Взаимосвязь между условием и заключением в нечетком правиле продукции в общем случае представляет собой некоторое бинарное нечеткое отношение на декартовом произведении универсальных множеств соответствующих лингвистических переменных. Этот подход и будет использоваться в дальнейшем для определения различных схем или методов нечеткого вывода на основе продукционных нечетких систем.

В общем случае для формального определения различных методов нечеткого вывода применительно к нечеткому правилу продукции рассмотрим два нечетких множества A и B, заданных соответственно на универсальных множествах U и V. При этом нечеткое множество A интерпретируется как условие некоторого нечеткого правила продукции, а нечеткое множество B – как заключение этого же правила.

Основная идея заключается в том, что нечеткое множество A можно рассматривать как унарное отношение на универсальном множестве U, а нечеткое множество B можно рассматривать как унарное отношение на

универсальном множестве *V*. В этом случае первое отношение определяется функцией принадлежности $\mu_A(u)$, а второе отношение – функцией принадлежности $\mu_B(v)$.

Теперь предположим, что некоторым образом определено бинарное нечеткое отношение на декартовом произведении универсальных множеств *U* и *V*: $Q = \{\mu_Q(u,v)/(u,v)\}$, где $u \in U$ и $v \in V$. Если дополнительно известны значения функции принадлежности множества *A* $\mu_A(u)$, то функция принадлежности $\mu_B(v)$ второго множества может быть определена в результате нечеткой композиции соответствующих нечетких отношений с использованием любой из приведенных ниже композиций.

Max-min композиция, или максиминная нечеткая свертка:

$$\mu_B(v) = \max_{x \in X}\{\min\{\mu_A(u), \mu_Q(<u,v>)\}\} \quad (4.8)$$

Max-prod композиция:

$$\mu_B(v) = \max_{x \in X}\{\mu_A(u) * \mu_Q(<u,v>)\} \quad (4.9)$$

Min-max композиция:

$$\mu_B(v) = \min_{x \in X}\{\max\{\mu_A(u), \mu_Q(<u,v>)\}\} \quad (4.10)$$

Max-max композиция:

$$\mu_B(v) = \max_{x \in X}\{\max\{\mu_A(v), \mu_Q(<u,v>)\}\} \quad (4.11)$$

Min-min композиция:

$$\mu_B(v) = \min_{x \in X}\{\min\{\mu_A(u), \mu_Q(<u,v>)\}\} \quad (4.12)$$

Max-average композиция:

$$\mu_B(v) = 0{,}5 * \max_{x \in X}\{\mu_A(u) + \mu_Q(<u,v>)\} \quad (4.13)$$

Sum-prod композиция:

$$\mu_B(v) = f(\sum_{x \in X}(\mu_A(u) * \mu_Q(<u,v>)), \quad (4.14)$$

где *f* – некоторая логистическая функция типа сигмоидной, которая ограничивает значения функции числом из интервала [0,1]. Этот метод композиции применяется в приложениях искусственных нейронных сетей для установления взаимосвязей между параллельными слоями в многослойных сетях.

4.2.4.2 Методы нечеткого вывода

По аналогии с обычными продукционными системами важным компонентом систем нечетких продукций является, так называемый, метод или схема вывода заключений на основе нечетких условий в базе правил нечетких продукций. Наиболее известными являются два метода вывода заключений: прямой и обратный, особенности которых рассматриваются ниже.

Прямой метод вывода заключений в системах нечетких продукций, называемый также методом нечеткого восходящего вывода или методом

прямой нечеткой цепочки рассуждений (fuzzy forward-chaining reasoning), основан на использовании нечеткого обобщения правила вывода modus ponens – FMP (fuzzy modus ponens). Согласно Л. Заде, суть нечеткого modus ponens заключается в следующем. Классическая импликация $A \supset B$ в правиле вывода modus ponens (MP) заменяется на правило нечеткой продукции: «ЕСЛИ *x* есть *A,* ТО *y* есть *B*», где *A* и *B* – нечеткие множества, а само правило нечеткой продукции представляет некоторое нечеткое отношение между переменными *x* и *y.* При этом $x \in X$ и $y \in Y$. Что касается посылки *A* правила MP, то она заменяется на нечеткое условие «*x* есть *A′*», где *A′* – нечеткое множество, отражающее знания о реальном значении переменной *x.* Объединение правила нечеткой продукции и нечеткого условия позволяет получить новую информацию о значении переменной *y* в форме: «*y* есть *B′*». При этом заключение по правилу FMP получается как функция принадлежности нечеткого множества *B′* на основе функции принадлежности условия *A′* и функции принадлежности нечеткой импликации как соответствующего нечеткого отношения с использованием одного из методов нечеткой композиции (4.8) – (4.14).

Применительно к системам нечетких продукций прямой метод вывода реализуется посредством преобразования отдельных фактов проблемной области в конкретные значения функций принадлежности условий нечетких продукций. После этого преобразования по одному из методов нечеткой композиции находятся значения функций принадлежности заключений правых частей по каждому из правил нечетких продукций. Эти значения функций принадлежности либо являются искомым результатом вывода, либо могут быть использованы в качестве дополнительных условий в рассматриваемой базе правил нечетких продукций. При этом правила, которые могут быть использованы для выполнения нечеткой композиции, также называют *активными.*

Процесс вывода прямым методом в системах нечетких продукций в общем случае может иметь рекурсивный (итеративный) характер. Он может быть остановлен либо в случае отсутствия активных правил нечетких продукций, либо в случае получения функции принадлежности заключения, которое является целевым в контексте решения исходной проблемы. В этом случае функция принадлежности заключения характеризует успех процесса вывода в системах нечетких продукций и решение поставленной проблемы.

Обратный метод вывода в продукционных системах, называемый также методом нечеткого нисходящего вывода или методом обратной нечеткой цепочки рассуждений (fuzzy backward-chaining reasoning), основан на использовании нечеткого обобщения правила вывода modus tollens – FMT (fuzzy modus tollens). Суть нечеткого modus tollens заключается в следующем. Классическая импликация $A \supset B$ в правиле вывода MT заменяется на правило нечеткой продукции: «ЕСЛИ *x* есть *A,* ТО *y* есть *B*», где *A* и *B* – нечеткие множества, а правило нечеткой продукции представляет некоторое нечеткое отношение между переменными *x* и *y,* при этом $x \in X$ и $y \in Y$, как

и в методе FMP. Заключение B заменяется нечетким заключением в форме «является ли y B'» или «y есть B'?». При этом нечеткое множество B' не равно нечеткому множеству B, используемому в заключении правила нечеткой продукции. Целью вывода методом обратной нечеткой цепочки рассуждений является установление истинности условия правила нечеткой продукции в форме: «является ли x A'» или «x есть A'?». В этом случае заключение по правилу FMT получается как функция принадлежности нечеткого множества A' на основе функции принадлежности заключения A' и функции принадлежности нечеткой импликации как соответствующего нечеткого отношения с использованием одного из методов нечеткой композиции (4.8) – (4.14).

Принципиальное различие между обратными методами вывода заключений в нечетких и обычных системах продукций заключается в том, что применительно к системам нечетких продукций функции принадлежности условий неизвестны и должны быть как-то заданы. Процесс обратного вывода в системах нечетких продукций начинается с подстановки отдельных интересующих нас значений функции принадлежности заключений в правые части соответствующих правил нечетких продукций, которые в этом случае становятся *активными.* После анализа каждого из активных правил находятся функции принадлежности условий, которые используются в этих правилах. Эти функции принадлежности условий принимаются в качестве подцелей, которые могут быть использованы в качестве функций принадлежности новых заключений в рассматриваемой базе правил нечетких продукций.

Процесс вывода обратным методом также имеет рекурсивный (итеративный) характер. Он может быть остановлен либо в случае отсутствия новых активных правил, либо в случае получения значений функций принадлежности условий, которые подтверждаются фактами проблемной области. Подобное подтверждение условий характеризует успех процесса вывода и справедливость значений функции принадлежности исходных заключений.

4.2.4.3 Этапы нечеткого логического вывода

Системы нечеткого вывода предназначены для преобразования значений входных переменных процесса управления в выходные переменные на основе использования нечетких правил продукционного вида. Для этого системы нечеткого вывода должны содержать базу правил нечетких продукций и реализовывать нечеткий вывод заключений на основе посылок или условий, представленных в форме нечетких лингвистических высказываний.

Таким образом, основными этапами нечеткого вывода являются:

1. Формирование базы правил систем нечеткого вывода.
2. Фаззификация входных переменных.
3. Агрегирование подусловий в нечетких правилах продукций.

4. Активизация или композиция подзаключений в нечетких правилах продукций.

5. Аккумулирование заключений нечетких правил продукций.

Ниже рассматриваются основные особенности каждого из этих этапов.

Формирование базы правил. База правил систем нечеткого вывода предназначена для формального представления эмпирических знаний или знаний экспертов в той или иной проблемной области. В системах нечеткого вывода используются правила нечетких продукций, в которых условия и заключения сформулированы в терминах нечетких лингвистических высказываний рассмотренных выше видов. Совокупность таких правил будем далее называть базами правил нечетких продукций. *База правил нечетких продукций* представляет собой конечное множество правил нечетких продукций, согласованных относительно используемых в них лингвистических переменных.

Согласованность правил относительно используемых лингвистических переменных означает, что в качестве условий и заключений правил могут использоваться только нечеткие лингвистические высказывания, при этом в каждом из таких высказываний должны быть определены функции принадлежности значений терм-множества для каждой из лингвистических переменных.

Входные и выходные лингвистические переменные. В системах нечеткого вывода лингвистические переменные, которые используются в нечетких высказываниях подусловий правил нечетких продукций, часто называют *входными лингвистическими переменными,* а переменные, которые используются в нечетких высказываниях подзаключений правил нечетких продукций, часто называют *выходными лингвистическими переменными.*

Таким образом, при задании или формировании базы правил нечетких продукций необходимо определить множество правил нечетких продукций $P=\{R_1, R_2,..., R_n\}$, множество входных лингвистических переменных $V=\{\beta_1,...,\beta_n\}$ и множество выходных лингвистических переменных $W=\{\omega_1,...,\omega_n\}$. Тем самым база правил нечетких продукций считается заданной, если заданы множества P, V и W.

Фаззификация. В контексте нечеткой логики под *фаззификацией* понимается не только отдельный этап выполнения нечеткого вывода, но и собственно процесс или процедура нахождения значений функций принадлежности нечетких множеств (термов) на основе обычных (не нечетких) исходных данных. Фаззификацию еще называют *введением нечеткости.*

Целью этапа фаззификации является установление соответствия между конкретным (обычно – численным) значением отдельной входной переменной системы нечеткого вывода и значением функции принадлежности соответствующего ей терма входной лингвистической переменной. После завершения этого этапа для всех входных переменных должны быть определены конкретные значения функций принадлежности по

каждому из лингвистических термов, которые используются в подусловиях базы правил системы нечеткого вывода.

Этап фаззификации считается законченным, когда будут найдены все значения $b_i = \mu(a_i)$ для каждого из подусловий всех правил, входящих в рассматриваемую базу правил системы нечеткого вывода. Это множество значений обозначим через $B=\{b_i'\}$. При этом, если некоторый терм α" лингвистической переменной β_i, не присутствует ни в одном из нечетких высказываний, то соответствующее ему значение функции принадлежности не находится в процессе фаззификации.

Агрегирование. Агрегирование представляет собой процедуру определения степени истинности условий по каждому из правил системы нечеткого вывода.

Формально процедура агрегирования выполняется следующим образом. До начала этого этапа предполагаются известными значения истинности всех подусловий системы нечеткого вывода, то есть множество значений $B=\{b_i'\}$. Далее рассматривается каждое из условий правил системы нечеткого вывода. Если условие правила представляет собой простое нечеткое высказывание, то степень его истинности равна соответствующему значению b_i. Если же условие состоит из нескольких подусловий, причем лингвистические переменные в подусловиях попарно не равны друг другу, то определяется степень истинности сложного высказывания на основе известных значений истинности подусловий. При этом значения b_i' используются в качестве аргументов соответствующих логических операций. Тем самым находятся количественные значения истинности всех условий правил системы нечеткого вывода.

Этап агрегирования считается законченным, когда будут найдены все значения b_k" для каждого из правил R_k, входящих в рассматриваемую базу правил P системы нечеткого вывода. Это множество значений обозначим через $B'' = \{b_1'', b_2'', ..., b_n''\}$.

Активизация. *Активизация* в системах нечеткого вывода представляет собой процедуру или процесс нахождения степени истинности каждого из подзаключений правил нечетких продукций. Активизация в общем случае во многом аналогична композиции нечетких отношений, но не тождественна ей.

Формально процедура активизации выполняется следующим образом. До начала этого этапа предполагаются известными значения истинности всех условий системы нечеткого вывода, то есть множество значений $B'' = \{b_1'', b_2'', ..., b_n''\}$ и значения весовых коэффициентов F для каждого правила. Далее рассматривается каждое из заключений правил системы нечеткого вывода. Если заключение правила представляет собой простое нечеткое высказывание, то степень его истинности равна алгебраическому произведению соответствующего значения b_i'' на весовой коэффициент F_i.

Если же заключение состоит из нескольких подзаключений, причем лингвистические переменные в подзаключениях попарно не равны друг другу, то степень истинности каждого из подзаключений равна

алгебраическому произведению соответствующего значения b_i'' на весовой коэффициент F_i. Таким образом, находятся все значения c_i степеней истинности подзаключений для каждого из правил R_k, входящих в рассматриваемую базу правил P системы нечеткого вывода. Это множество значений обозначим через $C=\{c_1, c_2,..., c_q\}$, где q – общее количество подзаключений в базе правил.

После нахождения множества $C=\{c_1, c_2,..., c_q\}$ определяются функции принадлежности каждого из подзаключений для рассматриваемых выходных лингвистических переменных. Для этой цели можно использовать один из методов, являющихся модификацией того или иного метода нечеткой композиции:

min-активизация:

$$\mu'(y) = \min\{c_i, \mu(y)\}; \quad (4.15)$$

prod-активизация:

$$\mu'(y) = c_i * \mu(y); \quad (4.16)$$

average-активизация:

$$\mu'(y) = 0{,}5 * (c_i + \mu(y)), \quad (4.17)$$

где $\mu(y)$ – функция принадлежности терма, который является значением некоторой выходной переменной ω_j, заданной на универсальном множестве Y.

Этап активизации считается законченным, когда для каждой из выходных лингвистических переменных, входящих в отдельные подзаключения правил нечетких продукций, будут определены функции принадлежности нечетких множеств и их значений, то есть совокупность нечетких множеств: $C_1, C_2,...,C_q$, где q – общее количество подзаключений в базе правил системы нечеткого вывода.

Аккумуляция. *Аккумуляция*, или аккумулирование, в системах нечеткого вывода представляет собой процедуру или процесс нахождения функции принадлежности для каждой из выходных лингвистических переменных множества $W = \{\omega_1, \omega_2,..., \omega_s\}$.

Цель аккумуляции заключается в том, чтобы объединить или аккумулировать все степени истинности заключений (подзаключений) для получения функции принадлежности каждой из выходных переменных. Причина необходимости выполнения этого этапа состоит в том, что подзаключения, относящиеся к одной и той же выходной лингвистической переменной, принадлежат различным правилам системы нечеткого вывода.

Формально процедура аккумуляции выполняется следующим образом. До начала этого этапа предполагаются известными значения истинности всех подзаключений для каждого из правил R_k, входящих в рассматриваемую базу правил P системы нечеткого вывода, в форме совокупности нечетких множеств: $C_1, C_2,...,C_q$, где q – общее количество подзаключений в базе правил. Далее последовательно рассматривается каждая из выходных лингвистических переменных $\omega_j \in W$ и относящиеся к ней нечеткие

множества: C_1, $C_2,...,C_q$. Результат аккумуляции для выходной лингвистической переменной ω_j определяется как объединение нечетких множеств C_1, $C_2,...,C_q$ по одной из следующих формул:

$$\mu_D(x) = \max\{\mu_A(x), \mu_B(x)\}, \quad (4.18)$$

$$\mu_D(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) * \mu_B(x), \quad (4.19)$$

$$\mu_D(x) = \min\{\mu_A(x) + \mu_B(x), 1\}. \quad (4.20)$$

Этап аккумуляции считается законченным, когда для каждой из выходных лингвистических переменных будут определены итоговые функции принадлежности нечетких множеств их значений, то есть совокупность нечетких множеств: $C'_1, C'_2, ...C'_S$, где S – общее количество выходных лингвистических переменных в базе правил системы нечеткого вывода.

Дефаззификация. *Дефаззификация* в системах нечеткого вывода представляет собой процедуру или процесс нахождения обычного (не нечеткого) значения для каждой из выходных лингвистических переменных множества $W = \{\omega_1, \omega_2, ..., \omega_s\}$.

Цель дефаззификации заключается в том, чтобы, используя результаты аккумуляции всех выходных лингвистических переменных, получить обычное количественное значение (crisp value) каждой из выходных переменных, которое может быть использовано специальными устройствами, внешними по отношению к системе нечеткого вывода.

Действительно, применяемые в современных системах управления устройства и механизмы способны воспринимать традиционные команды в форме количественных значений соответствующих управляющих переменных. Именно по этой причине необходимо преобразовать нечеткие множества в некоторые конкретные значения переменных. Поэтому дефаззификацию называют также приведением к четкости.

Формально процедура дефаззификации выполняется следующим образом. До начала этого этапа предполагаются известными функции принадлежности всех выходных лингвистических переменных в форме нечетких множеств: $C'_1, C'_2, ...C'_S$, где S – общее количество выходных лингвистических переменных в базе правил системы нечеткого вывода. Далее последовательно рассматривается каждая из выходных лингвистических переменных $\omega_j \in W$ и относящееся к ней нечеткое множество C'_j. Результат дефаззификации для выходной лингвистической переменной ω_j определяется в виде количественного значения $y \in R$, получаемого по одной из рассматриваемых ниже формул.

Этап дефаззификации считается законченным, когда для каждой из выходных лингвистических переменных будут определены итоговые количественные значения в форме некоторого действительного числа, то есть в виде y_1, y_2,..., y_s где S – общее количество выходных лингвистических переменных в базе правил системы нечеткого вывода.

Для выполнения численных расчетов на этапе дефаззификации могут быть использованы следующие формулы, получившие название *методов дефаззификации*.

Метод центра тяжести:

$$y = \frac{\int_{Min}^{Max} x * \mu(x) dx}{\int_{Min}^{Max} \mu(x) dx}, \quad (4.21)$$

где x – переменная, соответствующая выходной лингвистической переменной ω;

$\mu(x)$ – функция принадлежности нечеткого множества, соответствующего выходной лингвистической переменной ω;

Min и *Max* – левая и правая точки интервала носителя нечеткого множества рассматриваемой выходной лингвистической переменной ω.

Метод центра тяжести для одноточечных множеств:

$$y = \frac{\sum_{i=1}^{n} x_i * \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)}, \quad (4.22)$$

где n – число одноточечных (одноэлементных) нечетких множеств, каждое из которых характеризует единственное значение рассматриваемой выходной лингвистической переменной.

Метод центра площади:

$$\int_{Min}^{u} \mu(x) dx = \int_{u}^{Max} \mu(x) dx. \quad (4.23)$$

Метод левого модального значения:

$$y = \min\{x_m\}, \quad (4.24)$$

где x_m – модальное значение (мода) нечеткого множества, соответствующего выходной лингвистической переменной ω после аккумуляции, рассчитываемое по формуле $x_m = \arg \max_{x \in [a,b]} \{\mu(x)\}$

Метод правого модального значения:

$$y = \max\{x_m\}, \quad (4.25)$$

где x_m – модальное значение (мода) нечеткого множества, соответствующего выходной лингвистической переменной ω после аккумуляции, рассчитываемое аналогично предыдущему случаю.

4.2.4.4 Основные алгоритмы логического вывода

Рассмотренные выше этапы нечеткого вывода могут быть реализованы неоднозначным образом, поскольку включают в себя отдельные параметры, которые должны быть фиксированы или специфицированы. Тем самым

выбор конкретных вариантов параметров каждого из этапов определяет некоторый алгоритм, который в полном объеме реализует нечеткий вывод в системах правил нечетких продукций. К настоящему времени предложено несколько алгоритмов нечеткого вывода. Те из них, которые получили наибольшее применение в системах нечеткого вывода, рассматриваются ниже.

Алгоритм Мамдани является одним из первых, который нашел применение в системах нечеткого вывода. Он был предложен в 1975 г. английским математиком Е. Мамдани (Ebrahim Mamdani) в качестве метода для управления паровым двигателем. По своей сути этот алгоритм порождает рассмотренные выше этапы, поскольку в наибольшей степени соответствует их параметрам.

Формально алгоритм Мамдани может быть определен следующим образом.

– Формирование базы правил систем нечеткого вывода. Особенности формирования базы правил совпадают с рассмотренными выше при описании данного этапа.

– Фаззификация входных переменных. Особенности фаззификации совпадают с рассмотренными выше при описании данного этапа.

– Агрегирование подусловий в нечетких правилах продукций. Для нахождения степени истинности условий каждого из правил нечетких продукций используются парные нечеткие логические операции. Те правила, степень истинности условий которых отлична от нуля, считаются *активными* и используются для дальнейших расчетов.

– Активизация подзаключений в нечетких правилах продукций. Осуществляется по формуле (4.15), при этом для сокращения времени вывода учитываются только активные правила нечетких продукций.

– Аккумуляция заключений нечетких правил продукций. Осуществляется по формуле (4.18) для объединения нечетких множеств, соответствующих термам подзаключений, относящихся к одним и тем же выходным лингвистическим переменным.

Дефаззификация выходных переменных. Традиционно используется метод центра тяжести в форме (4.21) - (4.22) или метод центра площади (4.23).

Алгоритм Цукамото. Формально *алгоритм Цукамото* может быть определен следующим образом.

– Формирование базы правил систем нечеткого вывода. Особенности формирования базы правил совпадают с рассмотренными выше при описании данного этапа.

– Фаззификация входных переменных. Особенности фаззификации совпадают с рассмотренными выше при описании данного этапа.

– Агрегирование подусловий в нечетких правилах продукций. Для нахождения степени истинности условий всех правил нечетких продукций используются парные нечеткие логические операции. Те правила, степень

истинности условий которых отлична от нуля, считаются активными и используются для дальнейших расчетов.

– Активизация подзаключений в нечетких правилах продукций. Осуществляется аналогично алгоритму Мамдани по формуле (4.15), после чего находятся обычные (не нечеткие) значения всех выходных лингвистических переменных в каждом из подзаключений активных правил нечетких продукций. В этом случае значение выходной лингвистической переменной ω_i в каждом из подзаключений находится как решение уравнения:

$$c_i = \mu(w_j)\ (\forall i \in \{1,2,...,q\}), \tag{4.26}$$

где q – общее количество подзаключений в базе правил.

– Аккумуляция заключений нечетких правил продукций. Фактически отсутствует, поскольку расчеты осуществляются с обычными действительными числами ω_i.

– Дефаззификация выходных переменных. Используется модифицированный вариант в форме метода центра тяжести для одноточечных множеств:

$$y = \frac{\sum_{i=1}^{n} c_i * \varpi i}{\sum_{i=1}^{n} c_i}, \tag{4.27}$$

где n – общее количество активных правил нечетких продукций, в подзаключениях которых присутствует выходная лингвистическая переменная ω_i.

Алгоритм Сугено. Формально алгоритм Сугено, предложенный Сугено и Такаги, может быть определен следующим образом.

– Формирование базы правил систем нечеткого вывода. В базе правил используются только правила нечетких продукций в форме:

ПРАВИЛО <#>: ЕСЛИ «β_1 *есть* α'» И «β_2 *есть* α"» ТО «$w=\varepsilon$», (4.28)

здесь ε – некоторое действительное число.

– Фаззификация входных переменных. Особенности фаззификации совпадают с рассмотренными выше при описании данного этапа.

– Агрегирование подусловий в нечетких правилах продукций. Для нахождения степени истинности условий всех правил нечетких продукций, как правило, используется логическая операция min-конъюнкции. Те правила, степень истинности условий которых отлична от нуля, считаются *активными* и используются для дальнейших расчетов.

– Активизация подзаключений в нечетких правилах продукций. Во-первых, при использовании метода (4.15) находятся значения степеней истинности всех заключений правил нечетких продукций. Во-вторых, осуществляется расчет обычных (не нечетких) значений выходных переменных каждого правила. Расчет выполняется с использованием формулы для заключения (4.28), в которую вместо α' и α'' подставляются значения входных переменных до этапа фаззификации. Тем самым

определяются множество значений $C = \{c_1, c_2, ..., c_n\}$ и множество значений выходных переменных $W = \{\omega_1, \omega_2, ..., \omega_s\}$, где n – общее количество правил в базе правил.

– Аккумуляция заключений нечетких правил продукций. Фактически отсутствует, поскольку расчеты осуществляются с обычными действительными числами c_j.

– Дефаззификация выходных переменных. Используется модифицированный вариант в форме метода центра тяжести для одноточечных множеств (4.16).

4.2.4.5 Система нечеткого логического вывода

Архитектура нечеткого регулятора состоит из трёх компонентов: базы правил, интерфейсной части, аппарата нечёткого вывода.

Интерфейсная часть обеспечивает выполнение следующих функций:

– ввод исходных данных;

– создание, загрузка и сохранение базы правил нечетких продукций;

– ввод и редактирование переменных, термов, сфер и условий применимости, правил нечетких продукций на ограниченном подмножестве естественного языка;

– преобразование правила нечеткой продукции с естественно-языкового представления в предикатное;

– настройку методов логического вывода: активизации, аккумуляции и дефаззификации;

– отображение результатов нечёткого логического вывода в виде числовых данных и графиков функций принадлежности.

В базе правил хранятся множества входных и выходных лингвистических переменных с соответствующими им терм-множествами; множество четких переменных; множество продукционных правил, включающих условие применимости правила, посылку и заключение.

Аппарат нечеткого вывода выполняет нечеткий логический вывод по алгоритму Мамдани с выбранной точностью и обосновано выбранными методами активизации, аккумуляции и дефаззификации.

Система может работать в двух режимах: как автономная система и как встроенная. При сопряжении с аппаратом активации продукций система работает как встроенная. При этом предполагается, что все предварительные настройки должны быть выполнены заранее.

4.3 Выводы по главе

Модель аппарата активации, рассмотренная в работе, необходима для апробации сгенерированных методов естественно-языковой обработки научного текста. В принципе аппарат активации построен как настоящий модуль управления продукционными знаниями, поэтому его можно использовать и в реальной работе. Особенностью модуля является то, что в его состав входят как классический логический вывод, так и нечеткий. Для

поддержки единой линии представления методов нечеткие модели решения также представлены продукционными системами, только в данном случае нечеткими. Это определило реализацию нечеткого логического вывода, а именно применение методов нечеткого регулирования, которые хорошо вписываются в рассматриваемую технологию. Нечеткие модели решения применяются в тех случаях, когда с помощью классических продукций сложно решить проблему, в основном они используются для разрешения конфликтных ситуаций.

Модуль управления реализован с применением автоматного программирования, которое предоставляет широкие возможности по модификации и совершенствованию автоматных моделей с минимальным объемом прямого программирования.

Ядром модуля управления является система резолютивного логического вывода. Такие системы должны обладать эффективными алгоритмами поиска решений. В работе создание эффективной эвристики основано на эволюционной стратегии. Для каждого метода в процессе эволюции строится собственная модель автомата, настроенная на множество дизъюнктов этого метода. Поэтому система логического вывода, состоящая из множества конечных автоматов, модели которых построены на основе эволюций, а реализация – по технологии автоматного программирования, обладает эффективными алгоритмами поиска решений.

Заключение

В работе для реализации решения задач естественно-языковой обработки научных текстов и построения онтологий выбраны декларативные методы в виде систем продукций. Продукционные правила вида «если-то» – наиболее распространенный метод представления знаний в системах, основанных на знаниях. Правила обеспечивают естественный способ описания процессов в сложной изменяющейся внешней среде. В программах традиционного типа схема передачи управления предопределена в самой программе, а ветвление происходит только в заранее выбранных точках. Для интеллектуальных задач, где ветвление скорее норма, чем исключение, этот способ малоэффективен. В таких задачах правила дают возможность на каждом шаге решения оценить ситуацию и предпринять соответствующие действия.

Применение продукционных правил обеспечивает следующие преимущества: простота и высокое быстродействие; модульность – каждое правило описывает небольшой, относительно независимый блок знаний; удобство модификации – старые правила можно изменять и заменять на новые относительно независимо от других правил; ясность – знания в виде правил легко формулируются и воспринимаются экспертами; прозрачность – использование правил облегчает реализацию способности системы к объяснению принятых решений и полученных результатов; возможность постепенного наращивания – добавление правил в базу знаний происходит относительно независимо от других правил.

Кроме того, продукционные системы, как классические, так и нечеткие, содержащие аппарат логического вывода, отличает высокая степень общности правил обработки данных.

Второе значимое решение данной работы заключается в том, что для генерации моделей решения везде, где это возможно, использованы технологии генетического программирования. В работе представлен подход формализации и представления конструктивных знаний эксперта о методе решения задачи, который позволил автоматически генерировать системы продукций. Модели автоматов или преобразователей также генерируются посредством генетического программирования. Применение генетического программирования позволяет сделать перебор направленным, однако и в этом случае трудоемкость построения автоматов с требуемыми свойствами остается большой. Указанная проблема решается за счет учета специфики автоматов, для описания которой используется XML-описание предметной области прикладной задачи. В работе язык XML используется как средство внутренней и внешней коммуникации программных систем.

Третье решение, о котором следует упомянуть, это применение технологий автоматного программирования. Реализация моделей решения там, где это возможно, выполнена на основе применения автоматного программирования, которое в большой степени позволяет автоматизировать процесс получения корректного кода программ. Модификация таких

программ также осуществляется намного проще, чем при традиционном программировании.

Таким образом, в работе, когда имеется возможность представить решение в виде модели автомата или преобразователя, использованы технологии генетического и автоматного программирования в паре. Эти технологии применены при решении задач преобразования классических и нечетких продукционных правил из естественно-языкового представления в формулы предикатов первого порядка, преобразования предикатов во множество дизъюнктов и задачи линейного резолютивного вывода. В том случае, когда необходимо получить только модель решения задачи, использованы технологии генетического программирования (генерация систем продукций). И в случае, когда модель известна и необходимо разработать программное обеспечение, для неё использованы чистые технологии автоматного программирования (аппарат активации или модуль управления продукциями, процедуры постдействия продукций).

Реализация решения задач естественно-языковой обработки научного текста, осуществленная с применением технологий генетического и автоматного программирования, позволила создать технологию решения задач построения онтологий, отличающуюся почти полной автоматической обработкой.

Список литературы

1. Артемьева, И.Л. Метод построения многоуровневых онтологий сложноструктурированных предметных областей [Электронный ресурс] // Знания - онтологии - теории: труды Всерос. конф. – Новосибирск, 2007. – Режим доступа: http://www.iacp.dvo.ru/is/publications/Artemjeva.pdf.
2. Артемьева, И.Л. Многоуровневая онтология химии [Текст] / И.Л. Артемьева, Н.В. Рештаненко, В.А. Цветников // Знания - онтологии - теории: труды Всерос. конф. – Новосибирск: Ин-т математики, 2007. – Т.1. – С. 138-146.
3. Арутюнова, Н.Д. Язык цели [Текст] / Логический анализ языка: модели действия. – М.: Наука, 1992. – С. 14-30.
4. Асаи, К. Прикладные нечеткие системы [Текст] / К. Асаи, Д. Ватага, С. Иваи и др.; под ред. Тэрано Т. – М.: Мир, 1993. – С. 344.
5. Асанов, А.А. Исследовано в России [Электронный ресурс]. – Режим доступа: http://zhurnal.ape.relarn.ru/articles/2002/155.pdf.
6. Ахманова, О.С. Словарь лингвистических терминов [Текст] / М.: Советская энциклопедия, 1969. – 490с.
7. Аюшеева, Н.Н. Исследование и разработка моделей и методов поиска информационных образовательных ресурсов в электронной библиотеке: Дис. … канд. техн. наук: 05.13.11. – Улан-Удэ, 2004. – 218 с.
8. Багудина, Е. Г. Экономический словарь [Текст] / Е.Г.Багудина, и др.; отв. ред. А.И. Архипов. – М.: Изд-во Проспект, 2005. – 624 с.
9. Бахмутова, И.В. L-граммные азбуки для дешифровки знаменных песнопений [Текст] / И.В.Бахмутова, В.Д.Гусев, Т.Н.Титкова // Сиб. журн. индустр. матем. – Новосибирск: Ин-т математики, 1998. – Т. 1. – № 2 – С. 51–66.
10. Башмаков, А.И. Интеллектуальные информационные технологии: учеб. пособие [Текст] / А.И.Башмаков, И.А. Башмаков – М.: Изд-во МГТУ им. Н.Э.Баумана, 2005. – 304 с.
11. Беловольская, Л.А. Синтаксис словосочетания и простого предложения [Электронный ресурс]. – Режим доступа: ttp://www.philology.ru/linguistics2/belovolskaya-01.htm
12. Бин, Дж. XML для проектировщиков [Текст] /Дж. Бин – М.: Кудиц-Образ, 2004. – 256 с.
13. Бирюков, Б.В. О взглядах Г.Фреге на роль знаков и исчисления в познании [Текст] // Логическая структура научного знания: сб.науч. ст. – М.: Наука, 1965. – С. 91–108
14. Богатырев, М.Ю. Применение концептуальных графов в системах поддержки электронных библиотек [Электронный ресурс] /

М.Ю.Богатырев, В.Е.Латов, И.А. Столбовская // «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL'2007: труды 9-й Всерос. науч. конф. – Переславль-Залесский, 2007. – Режим доступа: http://rcdl2007.pereslavl.ru/ papers/paper_59_v2.pdf

15. Булыгина, Т.В. Грамматические и семантические категории и их связи [Текст] // Аспекты семантических исследований: сб.науч.ст. – М.: Наука, 1980. – С. 320-355.
16. Вагин, В. Н. Знание в интеллектуальных системах [Текст] // Новости искусственного интеллекта: журнал. – М.: Изд-во РАИИ, 2002. – №6 (54). – С. 8-18.
17. Вагин, В.Н. Достоверный и правдоподобный вывод в интеллектуальных системах [Текст] / В.Н.Вагин, Е.Ю.Головина, А.А. Загорянская; под ред. В.Н.Вагина, Д.А.Поспелова. – М.: Физматлит, 2004. – 704 с.
18. Валгина, Н.С. Современный русский язык [Текст] / Н.С.Валгина, Д. Э.Розенталь, М.И.Фомина – М.: Логос, 2002. – 528 с.
19. Вежбицкая, А. Русский язык [Текст] / Вежбицкая А. // Язык. Культура. Познание: Пер. с англ. – М.: Рус. словари, 1997. – С. 33-88.
20. Гаврилова, Т.А. Базы знаний интеллектуальных систем [Текст] / Т.А.Гаврилова, В.Ф.Хорошевский. – СПб: Питер, 2000. – 384 с.
21. Гаврилова, Т.А. Использование онтологий в системах управления знаниями [Электронный ресурс] / Т.А.Гаврилова. – Режим доступа: http://kmsoft.ru/ publications/library/authors/ use_ontology_in_suz.html
22. Гаврилова, Т.А. Формирование прикладных онтологий [Текст] / Т.А.Гаврилова // Труды XX нац. конф. по ИИ – КИИ-2006. – М.: Физматлит, 2006. – Т. 2.
23. Гак, В.Г. Высказывание и ситуация / В.Г.Гак // Проблемы структурной лингвистики. – М., 1973.
24. Гладков, Л.Д. Генетические алгоритмы [Текст] / Л.Д. Гладков, В.В. Курейчик, В.М. Курейчик; под. ред. В.М. Курейчика. – 2-е изд., испр. и доп. – М.: Физматлит, 2006. – 320 с.
25. Гладун, А.Я. Онтологии в корпоративных системах [Электронный ресурс] / А.Я.Гладун, Ю.В. Рогушина // Корпоративные системы: журнал. – М.: Комиздат, 2006 – №1– Режим доступа: http://www.management.com.ua/ims/ ims116.html?print
26. Гольдин, Д.А. Экспертные системы и продукционные правила в интеллектуальных комплексах распределения ресурсов автономных систем электроснабжения [Электронный ресурс] / Д.А. Гольдин, А.М.Чесноков. – Режим доступа: http://www.mtas.ru/Library/uploads/ 1206871348.pdf

27. Дальберг, И. Организация знаний: ее сфера и возможности [Текст] / И. Дальберг // Организация знаний: проблемы и тенденции: Программа и тез. докл. конф. – М., 1993. – С. 14.
28. Данилов, В.Р. Метод генетического программирования для генерации автоматов, представленных деревьями решений [Электронный ресурс] / В.Р.Данилов, А.А.Шалыто. – Режим доступа: http://is.ifmo.ru/download /2008-03-07-danilov.pdf.
29. Дейтел, П. Как программировать на XML [Текст] / П.Дейтел, П.Садху, Х.Дейтел – М.: Бином. Лаборатория знаний, 2008. – 944 с.
30. Добров, Б.В. Онтологии для автоматической обработки текстов: описание понятий и лексических значений [Текст] / Б.В.Добров, Н.В.Лукашевич; под ред. Н.И. Лауфер, А.С. Нариньяни, В.П. Селегея. // Компьютерная лингвистика и интеллектуальные технологии: труды Междунар. конф. "Диалог 2006". – М.: Изд-во РГГУ, 2006. – С. 138-143
31. Добров, Б.В. Онтологии и тезаурусы: учебно-методическое пособие [Текст] / Б.В.Добров, В.В.Иванов, Н.В.Лукашевич, В.Д.Соловьев. – Казань: Изд-во Казанского ГУ, 2006. – 190 с.
32. Евдокимова И.С. Методы и алгоритмы трансляции естественно-языковых запросов к базе данных в SQL-запросы: Дис. … канд. техн. наук: 05.13.11. – Улан-Удэ, 2004. – 172 с.
33. Ермаков, А.Е. Автоматизация онтологического инжиниринга в системах извлечения знаний из текста [Текст] / А.Е.Ермаков // Компьютерная лингвистика и интеллектуальные технологии: по материалам ежегодной Междунар. конф. «Диалог» (Бекасово, 4–8 июня 2008 г.). Вып. 7 (14).– М.: РГГУ, 2008. – С. 154–159.
34. Ершов, А.П. Терминологический словарь по основам информатики и вычислительной технике [Текст] / А.П.Ершов, Н.М.Шанский, А.П.Окунева, Н.В.Баско. – М.: Просвещение, 1991. – 159 с.
35. Жданов, Ю.Н. Динамическое смысловое представление текста. Язык, коммуникация и социальная среда [Электронный ресурс] / Ю.Н. Жданов; под ред. д.ф.н., профессора В.Б.Кашкина // Язык, коммуникация и социальная среда. – Воронеж: Изд-во ВГТУ, 2001. – Выпуск 1. – Режим доступа: http://tpl1999.narod.ru/WebLSE2001 /Zhdanov.htm
36. Искусственный интеллект: Справочник: В 3 кн. Кн.2. Модели и методы [Текст] / Под ред. Д.А.Поспелова. – М.: Радио и связь, 1990. – 304 с.
37. Казарина, В.И. Предложение. Текст. Речевое функционирование языковых единиц [Текст] / В.И.Казарина // Межвузовский сборник научных трудов. – Елец: Изд-во ЕГУ им. И. А. Бунина, 2002. – 214 с.
38. Ким, И.Е. Контроль и причастность в сфере человека и их отражение в

языке [Текст] / И.Е.Ким // Проблемы исторического языкознания и ментальности: сб. науч. статей. – Красноярск: Изд-во Краснояр. ун-та, 1999. – Вып. 3: Современное русское общественное сознание в зеркале вербализации. – С. 68-82.

39. Клещев, А.С. Онтология и модель онтологии предметной области "медицинская диагностика" [Текст] / А.С.Клещев, М.Ю.Черняховская, Ф.М.Москаленко. – Владивосток: Изд-во ИАПУ ДВО РАН, 2005. – 44 с.
40. Куршев, Е. П. Исследование методов извлечения информации из текстов с использованием автоматического обучения и реализация исследовательского прототипа системы извлечения информации [Текст] / Е.П.Куршев, Д.А.Кормалев, Е.А.Сулейманова, И.В. Трофимов // Математические методы распознавания образов: 13-я Всерос. конф.: сборник докладов. – М.: МАКС Пресс, 2007. – С.602-605.
41. Куршев, Е.П. Представление предметных знаний в системах интеллектуального анализа текста [Текст] / Е.П.Куршев, Е.А.Сулейманова // Программные системы: теория и приложения: Междунар. конф. – М.: Физматлит, 2006. – Т.1 – С. 379-390.
42. Ларичев, О. И. Выявление экспертных знаний [Текст] / О.И. Ларичев, А.И. Мечитов, Е.М. Мошкович, и др. – М.: Наука, 1996. – 128 с.
43. Леоненков, А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH [Текст] / А.В.Леоненков – СПб.: БХВ-Петербург, 2003. – 736 с.
44. Мелерович, А.М. Фразеологизмы в русской речи. Словарь [Текст] / А.М.Мелерович, В.М.Мокиенко – М.: Русские словари, 1997. – 864 с.
45. Мельников, Г.П. Основы терминоведения [Текст] / Г.П.Мельников. – М.: Изд-во ун-та дружбы народов, 1991. – 116с.
46. Минский, М. Фреймы для представления знаний [Текст] / М.Минский. – М.: Энергия, 1979. – 151 с.
47. Найханова, Л.В. Разработка конечного преобразователя продукционных правил на основе автоматного программирования [Текст] / Л.В. Найханова, Н.Б. Хаптахаева // Информационные технологии моделирования и управления: науч.-техн. журнал. – Воронеж: Научная книга, 2007. – Вып. 9(43).– С. 1082-1090.
48. Найханова, Л.В. Основные аспекты обработки поискового запроса на основе онтологического подхода [Текст] / Л.В. Найханова, Н.Н. Аюшеева // Вестник ВСГТУ: научный журнал. – Улан-Удэ: Изд-во ВСГТУ, 2007. – № 4. – С. 59-67.
49. Найханова, Л.В. Поиск по смыслу в текстовой информации на основе онтологического подхода [Текст] / Л.В. Найханова, Н.Н. Аюшеева // Теоретические и прикладные вопросы современных информационных технологий: мат-лы VIII Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во

ВСГТУ, 2007. – С. 164-171.

50. Найханова, Л.В. Формирование поискового образа текстового документа на основе онтологического подхода [Текст] / Л.В. Найханова, Н.Н.Аюшеева, Т.Н. Кушеева // Теоретические и прикладные вопросы современных информационных технологий: мат-лы Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2008. – С. 134-140.
51. Найханова, Л.В. Способ автоматического формирования системы продукций [Текст] / Л.В. Найханова, К.Ю. Васильцов // Теоретические и прикладные вопросы современных информационных технологий: мат-лы Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2008. – С. 152-158.
52. Найханова, Л.В. Формализация вопросно-ответных отношений в нечеткой модели оценивания результатов автоматизированного тестирования [Текст] / Л.В. Найханова, Данилова С.Д. // Проблемы информатизации региона: мат-лы 9-й Всерос. науч.-практ. конф. – Красноярск: ИПЦ КГТУ, 2005. – С. 142-147.
53. Найханова, Л.В. Конструкция знака концептуальных объектов и способ построения терминосистемы [Текст] / Л.В. Найханова // Материалы II Междунар. конф. по когнитивной науке. The Second Conference on Cognitive Science. – СПб., 2006. – С. 592-593.
54. Найханова, Л.В. Технология автоматического реферирования текста [Текст] / Л.В. Найханова, С.В. Машанова // Информационные системы и модели в научных исследованиях, промышленности и экологии: мат-лы Всерос. науч.-техн. конф. – Москва-Тула: Изд-во ТулГУ, 2007. – С. 69-70.
55. Найханова, Л.В. Основные аспекты понимания текста на естественном языке [Текст] / Л.В. Найханова // Теоретические и прикладные вопросы современных информационных технологий: мат-лы VII Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2006. – С. 126-130.
56. Найханова, Л.В. Основные аспекты построения онтологий верхнего уровня и предметной области [Текст] / Л.В. Найханова // Интернет-порталы: содержание и технологии: сб. науч. ст. Вып. 3 / [редкол.:А.Н.Тихонов (пред.) и др.]; ФГУ ГНИИ ИТТ «Информика». – М.: Просвещение, 2005. – С.452-479.
57. Найханова, Л.В. Построение семантической сети научного текста на основе аппарата расширенных семантических сетей [Текст] / Л.В. Найханова // Вестник ВСГТУ: Научный журнал. – Улан-Удэ: Изд-во ВСГТУ, 2008. – № 1. – С. 6-12.
58. Найханова, Л.В. Применение генетического и автоматного программирования в решении задач естественно-языковой обработки

монологического текста [Текст] / Л.В. Найханова // Управление созданием и развитием систем, сетей и устройств телекоммуникаций: труды науч.-практ. конф. – СПб.: НОЦ «Перспектива», 2008. – С. 292-309.

59. Найханова, Л.В. Применение методов нечеткого регулировании в соединении онтологий предметной области [Текст] / Л.В. Найханова // Программные продукты и системы: международный журнал. – Тверь: НИИ ЦПС, 2008. – №2. – С. 41-44.
60. Найханова, Л.В. Способ построения онтологической модели предметной области на основе ситуационного моделирования [Текст] / Л.В. Найханова // Вестник ВСГТУ: научный журнал. – Улан-Удэ: Изд-во ВСГТУ, 2007. – № 4. – С. 17-27.
61. Найханова, Л.В. Средства формализации методов естественно-языковой обработки информации [Текст] / Л.В. Найханова // Информационные системы и модели в научных исследованиях, промышленности и экологии: мат-лы Всерос. науч.-техн. конф. – Москва-Тула: Изд-во ТулГУ, 2007. – С. 67-68.
62. Найханова, Л.В. Структура словарных статей тезауруса [Текст] / Л.В. Найханова // Теоретические и прикладные вопросы современных информационных технологий: мат-лы Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2007. – С. 154-159.
63. Найханова, Л.В. Технология генетического программирования для генерации конечных преобразователей [Текст] / Л.В. Найханова // Системы управления и информационные технологии. Перспективные исследования: науч.-техн. журнал. – Воронеж: Научная книга, 2007. – №4.1 (30). – С. 174-178.
64. Найханова, Л.В. Технология решения задач естественно-языковой обработки монологического текста на основе использования генетического и автоматного программирования [Текст] / Л.В. Найханова // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление: научный журнал. – СПб.: Изд-во Политехн. ун-та, 2008. – № 2. – С.67-74
65. Найханова, Л.В. Соответствие лексико-семантических групп глаголов и основных категорий предложений, выбираемых для автоматического построения реферата естественно-языкового текста [Текст] / Л.В. Найханова, Н.Б. Хаптахаева, М.В. Кравченко // Теоретические и прикладные вопросы современных информационных технологий: мат-лы Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2004. – С. 63-66.
66. Найханова, Л.В. Методы и алгоритмы трансляции естественно-языковых запросов к базе данных в SQL-запросы: монография [Текст] / Л.В. Найханова, И.С. Евдокимова. – Улан-Удэ: Изд-во ВСГТУ, 2004. – 148 с.

67. Найханова, Л.В. Разрешение конфликтного множества графов зависимостей синтаксического анализатора на основе нейронной сети [Текст] / Л.В. Найханова, И.С. Евдокимова // Теоретические и прикладные вопросы современных информационных технологий: матлы IX Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2008. – С. 140-145.

68. Найханова, Л.В. Методика построения конечного преобразователя продукционных правил [Текст] / Л.В. Найханова, Н.Б. Хаптахаева // Системы управления и информационные технологии: науч.-техн. журнал. – Воронеж: Научная книга, 2008. – Вып. 1(31). – С. 83-88.

69. Найханова, Л.В. Структура источников знаний для создания тезауруса [Текст] / Л.В. Найханова, Н.Б. Хаптахаева // Теоретические и прикладные вопросы современных информационных технологий: Матлы Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2007. – С. 159-164.

70. Найханова, Л.В. Формирование библиотеки методов анализа научных текстов в виде систем продукций [Текст] / Л.В. Найханова, Н.Б. Хаптахаева // Открытое и дистанционное образование. – Томск: Изд-во ИДО ТГУ, 2005. – №4(20). – С. 13-24.

71. Найханова, Л.В. Построение семантической сети предметной области на основе извлечения знаний из научного текста [Текст] / Л.В. Найханова, Н.Б. Хаптахаева, Н.Н. Аюшеева // Известия высших учебных заведений. Поволжский регион. Технические науки. – Пенза: Изд-во «Информационно-издательский центр» ПензГУ, 2007. – № 4. – С. 51-61.

72. Найханова, Л.В. Механизм сопряжения генетического алгоритма с инструментальной системой UniMod [Текст] / Л.В. Найханова, Г.А. Хомонов // Информационные технологии моделирования и управления: науч.-техн. журнал. – Воронеж: Научная книга», 2007. – Вып. 1(44). – С. 86-91.

73. Найханова, Л.В. Применение генетического программирования при построении автоматной модели резолютивного вывода [Текст] / Л.В. Найханова, Г.А. Хомонов // Вестник СибГАУ им. академика М.Ф. Решетнева. – Красноярск: Изд-во СибГАУ, 2008. – Вып. 2(19). – С.78-82.

74. Найханова, Л.В. Технология интеграции генератора модели конечного преобразователя с инструментальной средой UniMod [Текст] / Л.В. Найханова, Г.А. Хомонов // Системы управления и информационные технологии: науч.-техн. журнал.– Воронеж: Научная книга, 2008. – Вып. 1(31). – С. 88-92.

75. Найханова, Л.В. Автоматическое реферирование научного текста на основе использования онтологического тезауруса [Текст] /

Л.В. Найханова, С.В. Машанова // Теоретические и прикладные вопросы современных информационных технологий: мат-лы Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2008. – С. 130-134.

76. Найханова, Л.В. Анализ научного текста и формирование категорийно-понятийного аппарата в виде терминосистемы [Текст] / Л.В. Найханова // Теоретические и прикладные вопросы современных информационных технологий: мат-лы Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2005. – С. 130-139.
77. Найханова, Л.В. Выделение словосочетаний для индексирования полнотекстовых документов [Текст] / Л.В. Найханова, Н.Н. Аюшеева, А.В. Шаманаев // Единая образовательная информационная среда: Проблемы и пути развития: мат-лы III Всерос. науч.-практ. конф.-выставки. – Омск: Изд-во ОмскГУ, 2004. – С. 283-285.
78. Найханова, Л.В. Генерация модели автоматического конечного преобразователя на основе генетического программирования [Текст] / Л.В. Найханова // Информационные технологии моделирования и управления: науч.-техн. журнал. – Воронеж: Научная книга, 2007. – Вып. 9(43). – С. 1046-1054.
79. Найханова, Л.В. Интерпретационная модель лингвистического транслятора [Текст] / Л.В. Найханова, И.С. Евдокимова // Теоретические и прикладные вопросы современных информационных технологий: Мат-лы VIII Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГТУ, 2007. – С. 173-180.
80. Нариньяни, А.С. Кентавр по имени ТЕОН: тезаурус+онтология [Текст] / А.С. Нариньяни // Междунар. семинар Диалог’2001 по компьютерной лингвистике и ее приложениям. – Аксаково, 2001. – Т. 1. – С. 184–188.
81. Нариньяни, А. С. ТЕОН-2: от тезауруса к онтологии и обратно [Текст] / А.С. Нариньяни // Компьютерная лингвистика и интеллектуальные технологии: междунар. семинар Диалог’2002 – М.: Наука, 2002. – Т. 1. – С. 307-313.
82. Никитина, С.Е. Семантический анализ языка науки [Текст] / С.Е. Никитина. – М.: Наука, 1987. – 143 с.
83. Осипов, Г.С. От ситуационного управления к прикладной семиотике [Текст] / Г.С. Осипов // Новости искусственного интеллекта. – М.: Изд-во РАИИ, 2002. – №6. – С. 3-7.
84. Осипов, Г.С. Построение моделей предметных областей. Неоднородные семантические сети [Текст] / Г.С. Осипов // Известия АН СССР. Техническая кибернетика. – М.: Изд-во АН СССР, 1990.– №5. – С.32-45.
85. Осипов, Г.С. Приобретение знаний интеллектуальными системами: Основы теории и технологии [Текст] / Г.С. Осипов – М.: Наука. Физматлит, 1997. – 112с.

86. Петушко, А.Г. Экспертная система на базе VP-expert [Электронный ресурс] // А.Г. Петушко, А.Г. Матусов, А.М. Ицков. – Режим доступа: http://prof9.narod.ru/library/lib010/ doc035.html.

87. Поликарпова, Н.И. Применение генетического программирования для реализации систем со сложным поведением [Электронный ресурс] / Н.И. Поликарпова, В.Н. Точилин, А.А. Шалыто. – Режим доступа: http://is.ifmo.ru/genalg/_polikarpova.pdf.

88. Поспелов, Д.А. Введение в прикладную семиотику [Текст] / Д.А. Поспелов, Г.С. Осипов // Новости искусственного интеллекта. – М.: Изд-во РАИИ, 2002. – № 6. – С. 28-35.

89. Поспелов, Д.А. Прикладная семиотика и искусственный интеллект [Текст] / Д.А. Поспелов // Программные продукты и системы. – Тверь: НИИ ЦПС, 1996. – №3. – С.10-13.

90. Поспелов, Д.А. Принципы ситуационного управления [Текст] / Д.А. Поспелов // Известия АН СССР. Техническая кибернетика. – М.: Изд-во АН ССР, 1971. – №2. – С.10-17.

91. Розенберг, Дж.М. Бизнес и менеджмент: терминологический словарь [Текст] / Дж.М. Розенберг. – М.: Инфра-М, 1997. – 400 с.

92. Розенберг, Дж.М. Инвестиции: терминологический словарь [Текст] / Дж.М. Розенберг. – М.: Инфра-М, 1997. – 400 с.

93. Рубашкин, В.Ш. Онтологии - проблемы и решения. Точка зрения разработчика [Электронный ресус] / В.Ш. Рубашкин. – Режим доступа: http://www.dialog-21.ru/dialog2007/ materials/html/74.htm

94. Сайт UniMod [Электронный ресурс]. – Режим доступа: http://unimod.sourceforge.net/.

95. Смирнов, А.В. Онтологии в системах искусственного интеллекта: способы построения и организации [Текст] / А.В. Смирнов, М.П. Пашкин, Н.Г. Шилов, Т.В. Левашова // Новости искусственного интеллекта. – М.: Изд-во РАИИ, 2002. – № 2. – С. 3-9.

96. Степанов, Ю.С. Семиотика [Текст] / Ю.С. Степанов. – М.: Наука, 1971 – 156 с.

97. Уемов, А.И. Вещи, свойства и отношения [Текст]. / А.И. Уемов – М., Изд-во АН СССР, 1963. – С. 5-33.

98. Хаптахаева, Н.Б. Модель МП-процессора, выполняющего преобразование естественно-языкового представления ядра продукции в формулу логики предикатов: Дис. … канд. техн. наук. – Улан-Удэ, 2005. – 201 с.

99. Хахалин, Г.К. Предметная онтология для понимания текстов геометрических задач [Электронный ресурс] // Компьютерная лингвистика и интеллектуальные технологии: междунар. семинар

Диалог'2008 – Режим доступа: http://www.dialog-21.ru/dialog2008/materials/html/Khakhalin.htm

100. Чень, Ч. Математическая логика и автоматическое доказательство теорем [Текст] / Ч. Чень, Р. Ли; под ред. С.Ю. Маслова. – М.: Наука, 1983.– 360 с.
101. Шалыто, А.А. Автоматно-ориентированное программирование [Текст] / А.А. Шалыто // Фундаментальные исследования в технических университетах: мат-лы IX Всерос. конф. по проблемам науки и высшей школы. – СПб.: Изд-во Политехн. ун-та, 2005. – С. 44-52.
102. Шалыто, А.А. Объектно-ориентированное программирование с явным выделением состояний [Текст] / А.А.Шалыто, Н.И. Туккель // Искусственный интеллект – 2002: мат-лы междунар. науч.-техн. конф. – Таганрог-Донецк: ТГРУ-ДИПИИ, 2002. – Т.1. – С. 198-202.
103. Яхно, Т. М. Системы продукций: структура, технология, применение [Текст] / Т. М. Яхно; под ред. Н. Н. Миренкова; АН СССР, Сиб. отд-ние, ВЦ. – Новосибирск: ВЦ СО АН СССР, 1990. – 127 с.
104. Ortega, Alfonso, Marina de la Cruz, Manuel Alfonseca. Christiansen Grammar Evolution: Grammatical Evolution With Semantics [Текст] / Ortega, Alfonso, Marina de la Cruz, Manuel Alfonseca. // IEEE Transactions on Evolutionary Computation. – 2007. –Vol. 11. – No. 1. – P. 77-90.
105. Brickley, D. Resource Description Framework (RDF) [Электронный ресурс] / D. Brickley, R.V. Guha // Schema Specification 1.0; 27 March 2000. – Режим доступа: http://www.w3.org/TR/2000/CR-rdf-schema-20000327
106. Buchheit, M. Decidable Reasoning in Terminological Knowledge Representation Systems [Текст] / M. Buchheit, F.M. Donini, A. Schaerf // Journal of Artificial Intelligence Research. – 1993. –Vol. 1. – P. 109-138.
107. Coello, E. Describing Reusable Problem-Solving Methods with a Method Ontology [Электронный ресурс] / E.Coello, G. Lapalme // Proceeding of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96). – Cateonia, Spain, 1997. – Режим доступа: http:/www.ksi.cpsc.ucalgary.ca/KAW/KAW96/ coelho/kaw.html.
108. Crow, L. Knowledge Engineering with Software Agent [Текст] / L. Crow, N.Shadbolt // Proceeding of the 1999 AAAI Spring Symposium on Intelligent Agent in Cyberspace. – Technical ReportSS-99-03, 1999. – P. 186-197.
109. Cui, Z. An Environment for Managing Enterprise Domain Ontology [Электронный ресурс] / Z. Cui, M. Cox, D.Jones // Chapter in Information Modeling for the New Millennium, IBSR. – 2000. – Режим доступа: http:/www.btexact.com/-project/-ibsr/publications.htm

110. Dalberg, I. Classification systems: information source on concepts and their relations [Текст] / I. Dalberg // Iinformation utilities. – Washington, 1974. – P. 69-72.

111. Davidson, D. События. Энциклопедия кругосвет. [Электронный ресурс] / D. Davidson. – Режим доступа: http://www.krugosvet.ru/articles/62/1006275/1006275a1.htm

112. Farquhar, A. Tools for Assembling Modular Ontologies in Ontolingua [Текст] / A. Farquhar, R. Fikes, Rice // Report of Knowledge Systems Laboratory. – Stanford University, USA, 1997. – P. 6.

113. Fernandez, M. METHONTOLOGY: From Ontological Art Toward Ontological Engineering [Текст] / M. Fernandez, A. Gomez-Perez, N.Juristo // Spring Symposium Series on Ontological Engineering AAAI-97. – Stanford: Stanford University, 1997.

114. Gangeni, A. An Overview of the ONIONS Project: Applying Ontologies to the Integration of Medical Terminologies [Текст] / A. Gangeni, D.M. Pisanelli, G. Steve // Data & Knowledge Engineering. – 1999. – V.31. – P. 183-220.

115. Gomez, F. Acquiring knowledge from encyclopedic texts [Текст] / F. Gomez, R. Hull, C. Segami // In Proc. of the ACL's 4th Conference on Applied Natural Language Processing, ANLP94. – Stuttgart, Germany, 1994. – P. 84-90.

116. Hendler, J. The DARPA Agent Markup Languages [Текст] / J. Hendler, D. McGuinness // IEEE Intelligent Systems, Trends and Controversies. – 2000. – P. 6-7.

117. Humphreys, B.L. The Unified Medical Language System: An Informatics Research Collaboration [Текст] / B.L. Humphreys, D.A.B. Lindberg, H.M. Schoolman, G.O. Barnett // J. Am. Medical Informatics Assoc. – 1998. – Vol. 5. – No. 1. – P. 1-11.

118. Jasper, R. A Framework for Understanding and Classifying Ontology Applications [Электронный ресурс] / R. Jasper, M.A. Uschold // Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management. – Voyager Inn, Banff, Alberta, Canada, 1999. – Режим доступа: http://sem.ucalgarv.ca/KSI/KAW/ KAW99/ papers.html.

119. Jordan, P.W. Learning Content Selection Rules for Generating Object Descriptions in Dialogue [Текст] / P.W. Jordan, M.A. Walker. – 2005. – Vol. 24. – P. 157-194.

120. (KA)2 Ontology OIL as result of the On-To-Knowledge- Project [Электронный ресурс]. – Режим доступа: http://www.ontoknowlege.org/oil/case-studies, 2001.

121. Karin Haenelt. Modelling natural language with finite automata [Электронный ресурс] / Karin Haenelt. – Режим доступа: http://kontext.fraunhofer.de/haenelt/papers/Theorietag14/HaeneltTheorietag14-L.pdf.

122. Kayed, A. Re-engineering Approach to Build Domain Ontologies. [Текст] / A. Kayed, R.M. Colomb // Proceedings of the First Asia-Pacific Conference on Web Intelligence (WI-2001). – Springer, Lecture Notes in Computer Science. – 2001. – No. 2198. – P. 464-472.

123. Kumar Chellapilla. Combining Mutation Operators in Evolutionary Programming [Текст] / Kumar Chellapilla. // IEEE transactions on Evolutionary Computation. – 1998. – Vol. 2. – No. 3. – P. 91-96.

124. Lankhorst, M. A genetic algorithm for induction of nondeterministic pushdown automata [Электронный ресурс] / M.Lankhorst // University of Groningen, Computer Science Report CS-R 9502. – 1995. – Режим доступа: http://www.lania.mx/ ~ccoello/lankhorst95.ps.gz

125. Lucas, S.M. Learning Deterministic Finite Automata with a Smart State Labeling Evolutionary alogarithm [Электронный ресурс]. / S.M. Lucas, T.J. Reynolds – Режим доступа: http://csdl2.computer.org/ persagen/DLAbsToc.jsp?resourcePath=/dl/trans/tp/&toc=comp/trans/tp/ 2005/07/i7toc.xml&DOI=10.1109/ TPAMI.2005.143

126. Lucas, S.M. Structuring chromosomes for context-free grammar evolution. – IEEE World Congress on Computational Intelligence [Текст] / S.M. Lucas // Proc. IEEE Int'l Conf. Evolutionary Computation. – 1994. – P. 130-135.

127. Mizoguchi Riichiro. Ontology Engineering Environment [Текст] / Mizoguchi Riichiro // Staab Steffen, Studer Rudi (eds). Handbook on Ontologies. – Berlin-Heidelberg: Springer-Verlag, 2004. – P. 275 – 295.

128. Montes-y-Gomez. Flexible Comparison of Conceptual Graphs [Текст] / Montes-y-Gomez, Gelbukh, Lopez-Lopez, Baeza-Yates // Proceedings of the 12th International Conference on Database and Expert Systems Applications table of contents – Springer-Verlag, 2001. –Vol. 2113. – P. 102-111.

129. Motta, E. A Principal Approach to the Constraction of a Task-specific Library of Problem Solving Components [Электронный ресурс] / E. Motta, Z. Zdrahal // A Proceeding of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management (KAW'98). – Inn, Banff, Alberta, Canada, 1998. – Режим доступа: http:/www.spuds.cpsc.ucalgary.ca /KAW/KAW98/Chandra/.

130. Naichanova, L.V. Knowledge presentation of inference system based on knowledge [Текст] / L.V. Naichanova // Information Technologies and Telecommunications in Education and Science (IT@I ES'2005): Materials of the international scientific conference / edited by A.N. Tikhonov (chair.) and others; SIIT@T Informika. – Moscow: VIZCOM, 2005. – P. 63–66.

131. Naidoo, A. The Induction of Finite Transducers Using Genetic Programming [Электронный ресурс] / A.Naidoo, N. Pillay. – Режим доступа: http://saturn.cs.unp.ac.za/~nelishiap/papers/ eurogp07.pdf.

132. Narayanan, S. Analysis and simulation of Web services [Текст] / S. Narayanan, S. McIlraith // Computer Networks: The International Journal of Computer and Telecommunications Networking. Volume 42 , Issue 5 (August 2003). Special issue: The Semantic Web: an evolution for a revolution. – New York, NY, USA Publisher Elsevier North-Holland, Inc., 2003. – P. 675-693.

133. Natalya F. Noy. Ontology Development 101: A Guide to Creating Your First Ontology. [Электронный ресурс] / Natalya F. Noy, Deborah L. McGuinness. // Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001. – Режим доступа: http://protege.stanford.edu/publications/ontology_development/ontology101.html

134. Oncina, J. The data driven approach applied to the OSTIA algorithm [Текст] / J. Oncina // Proceeding of the Fourth International Colloquium on Grammatical Inference; Lecture Notes in Computer Science. 1998. – Vol. 1433. – P. 50-56.

135. Ortiz-Boyer, D. CIXL2: A Crossover Operator for Evolutionary Algorithms Based on Population Features / D. Ortiz-Boyer, C.Herves-Martinez, N.Garcна-Pedrajas. – 2005. – Vol. 24. P. 1-48.

136. Petrovic, P. Comparing Finite State Automata Representation with GP-trees [Электронный ресурс] / Petrovic P. – Режим доступа: http://ii.fmph.uniba.sk/~petrovic/pub/fsatr.pdf.

137. Price, C. SNOMED clinical terms [Текст] / C. Price, K.Spackman // BJHCandIM-British Journal of Healthcare Computing and Information Management. 2001. – Vol. 17(3). – P. 27-31.

138. Schmidt, M. D. Coevolution of Fitness Predictors [Текст] / M.D. Schmidt, H.Lipson [Текст] // Evolutionary Computation, IEEE Transactions on. – Vol. PP, Issue 99. – P. 1-14.

139. Soderland, S. Wrap-Up: a Trainable Discourse Module for Information Extraction [Текст] / S.Soderland, W.Lehnert. – J. Artif. Intell. Res. (JAIR) 2, 1994. – Vol. 2. – P. 131-158.

140. Spackman, K.A. Compositional concept representation using SNOMED: towards further convergence of clinical terminologies. [Электронный ресурс] / K.A. Spackman, K.E. Campbell // Proc AMIA Symp, 1998:740-4. – Режим доступа: http://www.amia.org/pubs/symposia/D005143.PDF

141. The PICSEL Project [Электронный ресурс] // Laboratoire de Recherche Informatique of the Paris University, 2000. – Режим доступа: http:/www.lri.fr/LRI/iasi/theme.en.html.

142. The Upper CycOntology [Электронный ресурс]. – 2001. – Режим доступа: http://www.cyc.com/cyc-2-1/toc.html.

143. William M. Spears, An Analysis of Multi-Point Crossover. [Электронный ресурс] / M. William Spears, Kenneth A. De Jong. – Режим доступа:

http://www.cs.uwyo.edu/~wspears/ papers/foga90.pdf.

144. Wojciech Skut. Incremental Construction of Minimal Acyclic Sequental Transducers from Unsorted Data. [Электронный ресурс] / Wojciech Skut. – Режим доступа: http://www.cnts.ua.ac.be/clin2003/proc/09Skut.pdf.

ПРИЛОЖЕНИЕ А

Прототипы базовых знаков-фреймов

1. Фрейм-прототип концептуального объекта «Понятие»

```
<FRAME NAME = "Frame_#.xml" >
  <NAMETERM VALUE = "..." >
  <SLOT NAME = "NameCO" VALUE = "Понятие" >
  <SLOT NAME = "KindEntity" VALUE = "Материальный|Нематериальный" >
  <SLOT NAME = "definitions" >
    <SLOT NAME = "definition" VALUE = "..." >
    <SLOT NAME = "attachproc" VALUE = "FunDef" >
  <SLOT NAME = "propeties" >
    <SLOT NAME = "property" >
      <SLOT NAME = "nameprop" VALUE = "..." >
      <SLOT NAME = "link" FILE = "Frame_#.xml" >
    <SLOT NAME = "attachproc" VALUE = "FunProp" >
  <SLOT NAME = "actions" >
    <SLOT NAME = "action" >
      <SLOT NAME = "nameact" VALUE = "..." >
      <SLOT NAME = "link" FILE = "Frame_#.xml" >
    <SLOT NAME = "listattachproc" >
      <SLOT NAME = "attachproc" VALUE = "FunAct" >
      <SLOT NAME = "attachproc" VALUE = "ActivProc" >
  <SLOT NAME = "kvantrel" >
    <SLOT NAME = "synonyms" >
      <SLOT NAME = "synonym" >
        <SLOT NAME = "namesyn" VALUE = "..." >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
      <SLOT NAME = "attachproc" VALUE = "FunSyn" >
    <SLOT NAME = "correlates" >
      <SLOT NAME = "correlate" >
        <SLOT NAME = "namecorr" VALUE = "..." >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
      <SLOT NAME = "attachproc" VALUE = "FunCorr" >
  <SLOT NAME = "states" >
    <SLOT NAME = "state" >
      <SLOT NAME = "namestate" VALUE = "..." >
      <SLOT NAME = "link" FILE = "Frame_#.xml" >
    <SLOT NAME = "attachproc" VALUE = "FunState" >
  <SLOT NAME = "kvalitrel" >
    <SLOT NAME = "ClassKind" >
      <SLOT NAME = "Class" >
        <SLOT NAME = "nameclass" VALUE = "..." >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunClass" >
      <SLOT NAME = "Kind" >
        <SLOT NAME = "namekind" VALUE = "..." >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunKind" >
    <SLOT NAME = "WholePart" >
      <SLOT NAME = "Whole" >
        <SLOT NAME = "namewhole" VALUE = "..." >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunWhole" >
      <SLOT NAME = "Part" >
        <SLOT NAME = "namepart" VALUE = "..." >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunPart" >
    <SLOT NAME = "PresMethods" >
      <SLOT NAME = "PresMeth" >
        <SLOT NAME = "namemeth" VALUE = "..." >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
      <SLOT NAME = "attachproc" VALUE = "FunPresMeth" >
```

```
<SLOT NAME = "Styles" >
  <SLOT NAME = "Style" >
    <SLOT NAME = "namemeth" VALUE = "..." >
    <SLOT NAME = "link" FILE = "Frame_#.xml" >
  <SLOT NAME = "attachproc" VALUE = "FunStyle" >
<SLOT NAME = "metasign" >
  <SLOT NAME = "MetalangPresMethods" >
    <SLOT NAME = "MetalangPresMeth" >
      <SLOT NAME = "namemeth" VALUE = "..." >
      <SLOT NAME = "link" FILE = "Frame_#.xml" >
    <SLOT NAME = "attachproc" VALUE = "FunMetalangPres" >
  <SLOT NAME = "TermOthLang" >
    <SLOT NAME = "TermOthLang" >
      <SLOT NAME = "namemeth" VALUE = "..." >
      <SLOT NAME = "link" FILE = "Frame_#.xml" >
    <SLOT NAME = "attachproc" VALUE = "FunTOthLang" >
<SLOT NAME = "listattachproc" >
  <SLOT NAME = "attachproc" VALUE = "SearchTerm" >
  <SLOT NAME = "attachproc" VALUE = "GenThes" >
  <SLOT NAME = "attachproc" VALUE = "Visual" >
  <SLOT NAME = "attachproc" VALUE = "GenDictEntry" >
  <SLOT NAME = "attachproc" VALUE = "ActivProc" >
```

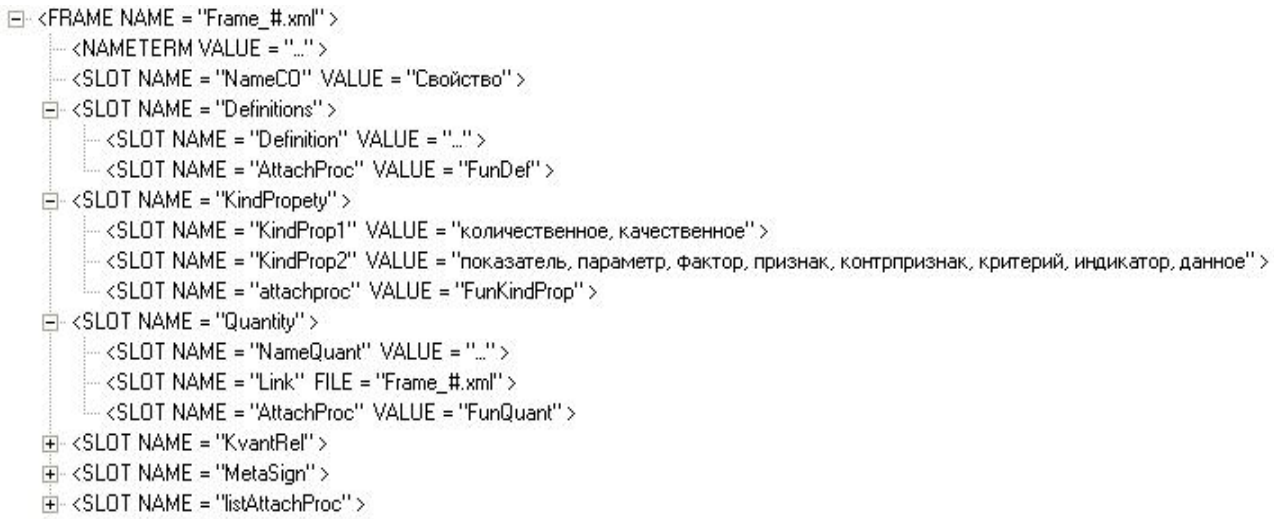
2. Фрейм-прототип концептуального объекта «Действие»

```
<FRAME NAME = "Frame_#.xml" >
  <NAMETERM VALUE = "..." >
  <SLOT NAME = "NameCO" VALUE = "Действие" >
  <SLOT NAME = "KindAct" VALUE = "процесс|функция|процедура|операция|способ|метод" >
  <SLOT NAME = "Definitions" >
    <SLOT NAME = "Definition" VALUE = "..." >
    <SLOT NAME = "AttachProc" VALUE = "FunDef" >
  <SLOT NAME = "Propeties" >
  <SLOT NAME = "SubjectObject" >
    <SLOT NAME = "SO" >
      <SLOT NAME = "TypeSO" VALUE = "субъект, объект" >
      <SLOT NAME = "NameTerm" VALUE = "..." >
      <SLOT NAME = "Link" FILE = "Frame_#.xml" >
    <SLOT NAME = "AttachProc" VALUE = "FunSO" >
  <SLOT NAME = "Tools" >
    <SLOT NAME = "Tool" >
      <SLOT NAME = "NameTool" VALUE = "..." >
      <SLOT NAME = "Link" FILE = "Frame_#.xml" >
    <SLOT NAME = "AttachProc" VALUE = "FunTool" >
  <SLOT NAME = "KvantRef" >
  <SLOT NAME = "Actions" >
    <SLOT NAME = "Action" >
      <SLOT NAME = "NameAct" VALUE = "..." >
      <SLOT NAME = "Link" FILE = "Frame_#.xml" >
    <SLOT NAME = "ListAttachProc" >
      <SLOT NAME = "AttachProc" VALUE = "FunAct" >
      <SLOT NAME = "AttachProc" VALUE = "ActivProc" >
  <SLOT NAME = "Events" >
    <SLOT NAME = "Event" >
      <SLOT NAME = "NameEvent" VALUE = "..." >
      <SLOT NAME = "Link" FILE = "Frame_#.xml" >
  <SLOT NAME = "MetaSign" >
  <SLOT NAME = "ListAttachProc" >
```

3. Фрейм-прототип концептуального объекта «Состояние»

4. Фрейм-прототип концептуального объекта «Событие»

5. Фрейм-прототип концептуального объекта «Свойство»



6. Фрейм-прототип концептуального объекта «Величина»

```
<FRAME NAME = "Frame_#.xml" >
    <NAMETERM VALUE = "..." >
    <SLOT NAME = "NameCO" VALUE = "Величина" >
    <SLOT NAME = "TypeQuantity" VALUE = "пространственная (длина, объём и т.д)|временная (длительность)|другие величины" >
  + <SLOT NAME = "Definitions" >
  - <SLOT NAME = "MeasurementScale" >
    - <SLOT NAME = "QualitativeMS" >
        <SLOT NAME = "SetValue1" VALUE = "..." >
        <SLOT NAME = "SetValue2" VALUE = "..." >
        <SLOT NAME = "AttachProc" VALUE = "FunQualitMS " >
    - <SLOT NAME = "QuantitativeMS" >
        <SLOT NAME = "MinValue" VALUE = "..." >
        <SLOT NAME = "MaxValue" VALUE = "..." >
      <SLOT NAME = "AttachProc" VALUE = "FunQuantitMS " >
  - <SLOT NAME = "InterOfValues" >
      <SLOT NAME = "MinNormValue" VALUE = "..." >
      <SLOT NAME = "MaxNormValue" VALUE = "..." >
      <SLOT NAME = "attachproc" VALUE = "FunIntOVal" >
    <SLOT NAME = "UnitMeasure" VALUE = "..." >
  + <SLOT NAME = "MetaSign" >
  + <SLOT NAME = "ListAttachProc" >
```

ПРИЛОЖЕНИЕ Б

Примеры систем продукций методов естественно-языковой обработки научного текста

1. Извлечение знаний о семантическом отношении «Целое-Часть»

1.1. Продукции для распознавания семантического отношения «Целое-Часть», расположенного в композитном словосочетании:

<*Продукционное правило* 1 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит <КомпСловоСоч>   k1          'И'
<КомпСловоСоч>   k1   содержит <ИССловоСоч>     s1          'И'
<КомпСловоСоч>   k1   содержит <ИССловоСоч>     s2          'И'
<ИССловоСоч>     s1   содержит <терм-спутникХ>  tx1         'И'
<ИССловоСоч>     s2   содержит <термин>         z1          'И'
<терм-спутникХ>  tx1  имеет    <значение>       ["часть"]   'И'
<термин>         z1   имеет    <характеристика> x1          'И'
<характеристика> x1   есть     <падеж>          c1          'И'
<падеж>          c1   имеет    <значение>       ["Род"]
ТО
<термин>         z1   имеет    <тип>     ["Часть"]   'И'
<ИССловоСоч>     s2   имеет    <тип>     ["Целое"]
```

<*Продукционное правило* 1 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,k1)        ∧ PAggr(Whole,k1,s1)    ∧
PAggr(Whole,k1,s2)         ∧ PAggr(Whole,s1,tx1)   ∧
PHier(Value,tx1,"часть")   ∧ PAggr(Whole,s2,z1)    ∧
PProp(Character,z1,x1)     ∧ PHier(Category,x1,c1) ∧
PHier(Value,c1,"Род"))     ⊃ (PHier(Category,z1,"Часть") ∧
PHier(Category,s2,"Целое"))
```

****

<*Продукционное правило* 2 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит <КомпСловоСоч>   k1          'И'
<КомпСловоСоч>   k1   содержит <ИССловоСоч>     s1          'И'
<КомпСловоСоч>   k1   содержит <ИССловоСоч>     s2          'И'
<ИССловоСоч>     s1   содержит <терм-спутникХ>  tx1         'И'
<ИССловоСоч>     s2   содержит <термин>         z1          'И'
<терм-спутникХ>  tx1  имеет    <значение>       ["элемент"] 'И'
<термин>         z1   имеет    <характеристика> x1          'И'
<характеристика> x1   есть     <падеж>          c1          'И'
<падеж>          c1   имеет    <значение>       ["Род"]
ТО
<термин>         z1   имеет    <тип>     ["Часть"]   'И'
<ИССловоСоч>     s2   имеет    <тип>     ["Целое"]
```

<*Продукционное правило* 2 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,k1)        ∧ PAggr(Whole,k1,s1)    ∧
PAggr(Whole,k1,s2)         ∧ PAggr(Whole,s1,tx1)   ∧
PHier(Value,tx1,"элемент")∧ PAggr(Whole,s2,z1)    ∧
PProp(Character,z1,x1)     ∧ PHier(Category,x1,c1) ∧
PHier(Value,c1,"Род"))     ⊃ (PHier(Category,z1,"Часть") ∧
PHier(Category,s2,"Целое"))
```

1.2. Продукции для распознавания семантического отношения «Целое-Часть» в предложении, содержащем список понятий

<*Продукционное правило* 3 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <КомпСловоСоч>   k1       'И'
<предложение>    p1   содержит  <СемОтношение>   r1       'И'
<предложение>    p1   содержит  <список>         q1       'И'
<КомпСловоСоч>   k1   содержит  <ИССловоСоч>     s1       'И'
<КомпСловоСоч>   k1   содержит  <ИССловоСоч>     s2       'И'
<ИССловоСоч>     s2   содержит  <термин>         z1       'И'
<СемОтношение>   r1   содержит  <глагол>         v1       'И'
<глагол>         v1   имеет     <значение>       ["включает"]'И'
<СемОтношение>   r1   содержит  <терм-спутникR>  tr1      'И'
<терм-спутникR>  tr1  имеет     <значение>       ["в себя"]  'И'
<список>         q1   содержит  <ЭлементСписка>  e1       'И'
<ИССловоСоч>     s1   имеет     <характеристика> x1       'И'
<характеристика> x1   есть      <падеж>          c1       'И'
<термин>         z1   имеет     <характеристика> x2       'И'
<характеристика> x2   есть      <падеж>          c2       'И'
<ЭлементСписка>  e1   есть      <ИССловоСоч>     s3       'И'
<ИССловоСоч>     s3   имеет     <характеристика> x3       'И'
<характеристика> x3   есть      <падеж>          c3       'И'
<падеж>          c1   имеет     <значение>       ["Им"]  'И'
<падеж>          c2   имеет     <значение>       ["Род"] 'И'
<падеж>          c3   имеет     <значение>       ["Род"] 'И'
<СемОтношение>   r1   имеет     <индекс>         i1       'И'
<список>         q1   имеет     <индекс>         (i1+1)
ТО
<ИССловоСоч>     s1   имеет       <тип>          ["Целое"]   'И'
<список>         q1   имеет       <тип>          ["Части"]   'И'
<СемОтношение>   r1   относится к <категория>    ["ЦелоеЧасть"]
```

<*Продукционное правило* 3 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,k1)   ∧ PAggr(Whole,p1,r1)    ∧
PAggr(Whole,p1,q1)    ∧ PAggr(Whole,k1,s1)    ∧
PAggr(Whole,k1,s2)    ∧ PAggr(Whole,s2,z1)    ∧
PAggr(Whole,r1,v1)    ∧ PHier(Value,v1,"включает")∧
PAggr(Whole,r1,tr1)   ∧ PHier(Value,tr1,"в себя") ∧
PAggr(Whole,q1,e1)    ∧ PProp(Character,s1,x1)  ∧
PHier(Category,x1,c1)∧ PProp(Character,z1,x2)  ∧
PHier(Category,x2,c2)∧ PHier(Category,e1,s3)   ∧
PProp(Character,s3,x3)∧ PHier(Category,x3,c3)   ∧
PHier(Value,c1,"Им") ∧ PHier(Value,c2,"Род")   ∧
PHier(Value,c3,"Род")∧ PHier(Index,r1,i1)
PHier(Index,q1,(i1+1))) ⊃ (PHier(Category,q1,"Части") ∧
PHier(Category,s1,"Целое") ∧   PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 4 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <термин>         z1           'И'
<предложение>    p1   содержит  <СемОтношение>   r1           'И'
<предложение>    p1   содержит  <признак>        h1           'И'
<предложение>    p1   содержит  <список>         q1           'И'
<СемОтношение>   r1   содержит  <глагол>         v1           'И'
<глагол>         v1   имеет     <значение>       ["состоит"] 'И'
<СемОтношение>   r1   содержит  <терм-спутникR>  tr1          'И'
<терм-спутникR>  tr1  имеет     <значение>       ["из частей"]  'И'
<список>         q1   содержит  <ЭлементСписка>  e1           'И'
```

```
<признак>        h1   имеет     <значение>       [':']        'И'
<ЭлементСписка>  e1   имеет     <характеристика> x1           'И'
<характеристика> x1   есть      <ЧастьРечи>      c1           'И'
<ЧастьРечи>      c1   имеет     <значение>       ["Сущ"]      'И'
<термин>         z1   имеет     <индекс>         i1           'И'
<СемОтношение>   r1   имеет     <индекс>         (i1+1)       'И'
<признак>        h1   имеет     <индекс>         (i1+2)       'И'
<список>         q1   имеет     <индекс>         (i1+3)
ТО
<термин>         z1   имеет       <тип>       ["Целое"]   'И'
<список>         q1   имеет       <тип>       ["Части"]   'И'
<СемОтношение>   r1   относится к <категория> ["ЦелоеЧасть"]
```

<*Продукционное правило* 4 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,z1)     ∧   PAggr(Whole,p1,r1)          ∧
PAggr(Whole,p1,q1)      ∧   PAggr(Whole,p1,h1)          ∧
PAggr(Whole,r1,v1)      ∧   PHier(Value,v1,"состоит из")∧
PAggr(Whole,r1,tr1)     ∧   PHier(Value,tr1,"частей")   ∧
PAggr(Whole,q1,e1)      ∧   PProp(Character,e1,x1)      ∧
PHier(Category,x1,c1)   ∧   PHier(Value,c1,"Сущ")       ∧
PHier(Value,h1,":")     ∧   PHier(Index,z1,i1)          ∧
PHier(Index,r1,(i1+1))  ∧   PHier(Index,h1,(i1+2))      ∧
PHier(Index,q1,(i1+3))) ⊃   (PHier(Category,q1,"Части") ∧
PHier(Category,z1,"Целое")  ∧ PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 5 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <термин>         z1              'И'
<предложение>    p1   содержит  <СемОтношение>   r1              'И'
<предложение>    p1   содержит  <список>         q1              'И'
<СемОтношение>   r1   содержит  <глагол>         v1              'И'
<глагол>         v1   имеет     <значение>       ["включает"]    'И'
<СемОтношение>   r1   содержит  <терм-спутникR>  tr1             'И'
<терм-спутникR>  tr1  имеет     <значение>       ["составных элементов"]'И'
<список>         q1   содержит  <ЭлементСписка>  e1              'И'
<ЭлементСписка>  e1   имеет     <характеристика> x1              'И'
<характеристика> x1   есть      <ЧастьРечи>      c1              'И'
<ЧастьРечи>      c1   имеет     <значение>       ["Сущ"]         'И'
<термин>         z1   имеет     <индекс>         i1              'И'
<СемОтношение>   r1   имеет     <индекс>         (i1+1)          'И'
<список>         q1   имеет     <индекс>         (i1+2)
ТО
<термин>         z1   имеет       <тип>          ["Целое"]   'И'
<список>         q1   имеет       <тип>          ["Части"]   'И'
<СемОтношение>   r1   относится к <категория>  ["ЦелоеЧасть"]
```

<*Продукционное правило* 5 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,z1)   ∧ PAggr(Whole,p1,r1)             ∧
PAggr(Whole,p1,q1)    ∧ PAggr(Whole,r1,v1)             ∧
PAggr(Whole,r1,tr1)   ∧ PHier(Value,v1,"включает")     ∧
PHier(Value,tr1,"составных элементов")                 ∧
PAggr(Whole,q1,e1)    ∧ PProp(Character,e1,x1)         ∧
PHier(Category,x1,c1) ∧ PHier(Value,c1,"Сущ")          ∧
PHier(Index,z1,i1)    ∧ PHier(Index,r1,(i1+1))         ∧
PHier(Index,q1,(i1+2))) ⊃ (PHier(Category,q1,"Части")  ∧
PHier(Category,z1,"Целое") ∧   PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 6 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит <термин>          z1       'И'
<предложение>    p1   содержит <СемОтношение>    r1       'И'
<предложение>    p1   содержит <термин>          t1       'И'
<СемОтношение>   r1   содержит <глагол>          v1       'И'
<глагол>         v1   имеет    <значение>        ["состоят"] 'И'
<СемОтношение>   r1   содержит <терм-спутникR>   tr1      'И'
<терм-спутникR>  tr1  имеет    <значение>        ["из"]   'И'
<термин>         t1   имеет    <характеристика>  x1       'И'
<характеристика> x1   есть     <падеж>           c1       'И'
<падеж>          c1   имеет    <значение>        ["Род"]  'И'
<термин>         z1   имеет    <индекс>          i1       'И'
<СемОтношение>   r1   имеет    <индекс>          (i1+1)   'И'
<термин>         t1   имеет    <индекс>          (i1+2)
ТО
<термин>         z1   имеет       <тип>          ["Целое"]   'И'
<термин>         t1   имеет       <тип>          ["Часть"]   'И'
<СемОтношение>   r1   относится к <категория>    ["ЦелоеЧасть"]
```

<*Продукционное правило* 6 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,z1)        ∧ PAggr(Whole,p1,r1)          ∧
PAggr(Whole,p1,t1)         ∧ PAggr(Whole,r1,v1)          ∧
PHier(Value,v1,"состоят")  ∧ PAggr(Whole,r1,tr1)         ∧
PHier(Value,tr1,"из")      ∧ PProp(Character,t1,x1)      ∧
PHier(Category,x1,c1)      ∧ PHier(Value,c1,"Род")       ∧
PHier(Index,z1,i1))        ∧ PHier(Index,r1,(i1+1))      ∧
PHier(Index,t1,(i1+2)))    ⊃ (PHier(Category,t1,"Часть") ∧
PHier(Category,z1,"Целое")∧ PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 7 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит <ИССловоСоч>      s1           'И'
<предложение>    p1   содержит <СемОтношение>    r1           'И'
<предложение>    p1   содержит <признак>         h1           'И'
<предложение>    p1   содержит <список>          q1           'И'
<ИССловоСоч>     s1   содержит <термин>          z1           'И'
<ИССловоСоч>     s1   содержит <Терм-спутникX>   tx1          'И'
<список>         q1   содержит <ЭлементСписка>   e1           'И'
<СемОтношение>   r1   содержит <глагол>          v1           'И'
<глагол>         v1   имеет    <значение>        ["включает"] 'И'
<СемОтношение>   r1   содержит <терм-спутникR>   tr1          'И'
<терм-спутникR>  tr1  имеет    <значение>        ["в себя"]   'И'
<термин>         z1   имеет    <характеристика>  x1           'И'
<характеристика> x1   есть     <падеж>           c1           'И'
<Терм-спутникX>  tx1  имеет    <значение>        ["традиционный состав"] 'И'
<Терм-спутникX>  tx1  имеет    <характеристика>  x2           'И'
<характеристика> x2   есть     <падеж>           c2           'И'
<признак>        h1   имеет    <значение>        [':']        'И'
<ЭлементСписка>  e1   есть     <ИССловоСоч>      s2           'И'
<ИССловоСоч>     s2   имеет    <характеристика>  x3           'И'
<характеристика> x3   есть     <падеж>           c3           'И'
<падеж>          c1   имеет    <значение>        ["Род"]      'И'
<падеж>          c2   имеет    <значение>        ["Им"]       'И'
<падеж>          c3   имеет    <значение>        ["Род"]      'И'
<ИССловоСоч>     s1   имеет    <индекс>          i1           'И'
<СемОтношение>   r1   имеет    <индекс>          (i1+1)       'И'
<признак>        h1   имеет    <индекс>          (i1+2)       'И'
<список>         q1   имеет    <индекс>          (i1+3)
ТО
```

<термин> z_1 имеет <тип> [“Целое”] ‘И’
<список> q_1 имеет <тип> [“Части”] ‘И’
<СемОтношение> r_1 относится к <категория> [“ЦелоеЧасть”]

<*Продукционное правило* 1 на языке логики предикатов первого порядка>

(PAggr(Whole,p1,s1) ∧ PAggr(Whole,p1,r1) ∧
PAggr(Whole,p1,q1) ∧ PAggr(Whole,p1,h1) ∧
PAggr(Whole,s1,z1) ∧ PAggr(Whole,s1,tx1) ∧
PAggr(Whole,r1,v1) ∧ PHier(Value,v1,“включает”)∧
PAggr(Whole,r1,tr1) ∧ PHier(Value,tr1,“в себя”) ∧
PProp(Character,z1,x1) ∧ PHier(Category,x1,c1) ∧
PHier(Value,tx1,“традиционный состав”) ∧
PProp(Character,tx1,x2) ∧ PHier(Category,x2,c2) ∧
PHier(Value,h1,“:”) ∧ PAggr(Whole,q1,e1) ∧
PHier(Category,e1,s2) ∧ PProp(Character,s2,x3) ∧
PHier(Category,x3,c3) ∧ PHier(Value,c1,“Род”) ∧
PHier(Value,c1,“Им”) ∧ PHier(Value,c1,“Род”) ∧
PHier(Index,s1,i1) ∧ PHier(Index,r1,(i1+1)) ∧
PHier(*Index*,*h*1,(*i*1+2)) ∧ *PHier*(*Index*,*q*1,(*i*1+3))) ⊃
(PHier(Category,q1,“Части”) ∧ PHier(Category,z1,“Целое”)∧
PHier(Category,r1,“ЦелоеЧасть”))

****

<*Продукционное правило* 8 на ограниченном подмножестве естественного языка>

ЕСЛИ
<предложение> p_1 содержит <термин> z_1 ‘И’
<предложение> p_1 содержит <СемОтношение> r_1 ‘И’
<предложение> p_1 содержит <список> q_1 ‘И’
<СемОтношение> r_1 содержит <глагол> v_1 ‘И’
<глагол> v_1 имеет <значение> [“состоит”] ‘И’
<СемОтношение> r_1 содержит <терм-спутникR> tr_1 ‘И’
<терм-спутникR> tr_1 имеет <значение> [“из”] ‘И’
<список> q_1 содержит <ЭлементСписка> e_1 ‘И’
<термин> z_1 имеет <характеристика> x_1 ‘И’
<характеристика> x_1 есть <падеж> c_1 ‘И’
<ЭлементСписка> e_1 имеет <характеристика> x_2 ‘И’
<характеристика> x_2 есть <ЧастьРечи> c_2 ‘И’
<ЭлементСписка> e_1 имеет <характеристика> x_3 ‘И’
<характеристика> x_3 есть <падеж> c_3 ‘И’
<падеж> c_1 имеет <значение> [“Им”] ‘И’
<падеж> c_3 имеет <значение> [“Род”] ‘И’
<ЧастьРечи> c_2 имеет <значение> [“Сущ”] ‘И’
<термин> z_1 имеет <индекс> i_1 ‘И’
<СемОтношение> r_1 имеет <индекс> (i_1+1) ‘И’
<список> q_1 имеет <индекс> (i_1+2)
ТО
<термин> z_1 имеет <тип> [“Целое”] ‘И’
<список> q_1 имеет <тип> [“Части”] ‘И’
<СемОтношение> r_1 относится к <категория> [“ЦелоеЧасть”]

<*Продукционное правило* 8 на языке логики предикатов первого порядка>

(PAggr(Whole,p1,z1) ∧ PAggr(Whole,p1,r1) ∧
PAggr(Whole,p1,q1) ∧ PAggr(Whole,r1,v1) ∧
PAggr(Whole,r1,tr1) ∧ PHier(Value,v1,“состоит”) ∧
PHier(Value,tr1,“из”) ∧ PAggr(Whole,q1,e1) ∧
PProp(Character,z1,x1) ∧ PHier(Category,x1,c1) ∧
PProp(Character,e1,x2) ∧ PHier(Category,x2,c2) ∧
PProp(Character,e1,x3) ∧ PHier(Category,x3,c3) ∧
PHier(Value,c1,“Им”) ∧ PHier(Value,c2,“Род”) ∧

```
PHier(Value,c3,"Сущ")   ∧ PHier(Index,z1,i1)      ∧
PHier(Index,r1,(i1+1))  ∧ PHier(Index,q1,(i1+2))) ⊃
(PHier(Category,q1,"Части") ∧ PHier(Category,z1,"Целое") ∧
PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 9 на ограниченном подмножестве естественного языка>

ЕСЛИ

<предложение>	p_1	содержит	<КомпСловоСоч>	k_1	'И'
<предложение>	p_1	содержит	<СемОтношение>	r_1	'И'
<предложение>	p_1	содержит	<ИССловоСоч>	s_3	'И'
<КомпСловоСоч>	k_1	содержит	<ИССловоСоч>	s_1	'И'
<КомпСловоСоч>	k_1	содержит	<ИССловоСоч>	s_2	'И'
<ИССловоСоч>	s_1	содержит	<терм-спутникХ>	tx_1	'И'
<СемОтношение>	r_1	содержит	<глагол>	v_1	'И'
<глагол>	v_1	имеет	<значение>	["является"]	'И'
<ИССловоСоч>	s_2	содержит	<термин>	z_1	'И'
<терм-спутникХ>	tx_1	имеет	<значение>	["частью"]	'И'
<терм-спутникХ>	tx_1	имеет	<характеристика>	x_1	'И'
<характеристика>	x_1	есть	<падеж>	c_1	'И'
<термин>	z_1	имеет	<характеристика>	x_2	'И'
<характеристика>	x_2	есть	<падеж>	c_2	'И'
<падеж>	c_1	имеет	<значение>	["Тв"]	'И'
<падеж>	c_2	имеет	<значение>	["Род"]	'И'
<КомпСловоСоч>	k_1	имеет	<индекс>	i_1	'И'
<СемОтношение>	r_1	имеет	<индекс>	(i_1+1)	'И'
<ИССловоСоч>	s_3	имеет	<индекс>	(i_1+2)	

ТО

<термин>	z_1	имеет	<тип>	["Целое"]	'И'
<ИССловоСоч>	s_3	имеет	<тип>	["Часть"]	'И'
<СемОтношение>	r_1	относится к	<категория>	["ЦелоеЧасть"]	

<*Продукционное правило* 9 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,k1)     ∧ PAggr(Whole,p1,r1)          ∧
PAggr(Whole,p1,s3)      ∧ PAggr(Whole,k1,s1)          ∧
PAggr(Whole,k1,s2)      ∧ PAggr(Whole,s1,tx1)         ∧
PAggr(Whole,r1,v1)      ∧ PHier(Value,v1,"является")  ∧
PAggr(Whole,s2,z1)      ∧ PHier(Value,tx1,"частью")   ∧
PProp(Character,tx1,x1) ∧ PHier(Category,x1,c1)       ∧
PProp(Character,z1,x2)  ∧ PHier(Category,x2,c2)       ∧
PHier(Value,c1,"Тв")    ∧ PHier(Value,c2,"Род")       ∧
PHier(Index,k1,i1)      ∧ PHier(Index,r1,(i1+1))      ∧
PHier(Index,s3,(i1+2))) ⊃ (PHier(Category,s3,"Часть") ∧
PHier(Category,z1,"Целое")  ∧ PHier(Category,r1,"ЦелоеЧасть"))
```

3. Продукции для распознавания семантического отношения «Целое-Часть» в предложении, в котором заголовочный термин выделен как особый

<*Продукционное правило* 10 на ограниченном подмножестве естественного языка>

ЕСЛИ

<предложение>	p_1	содержит	<КомпСловоСоч>	k_1	'И'
<предложение>	p_1	содержит	<СемОтношение>	r_1	'И'
<предложение>	p_1	содержит	<лексема>	l_1	'И'
<КомпСловоСоч>	k_1	содержит	<ИССловоСоч>	s_1	'И'
<КомпСловоСоч>	k_1	содержит	<ИССловоСоч>	s_2	'И'
<ИССловоСоч>	s_1	содержит	<терм-спутникХ>	tx_1	'И'
<СемОтношение>	r_1	содержит	<глагол>	v_1	'И'
<глагол>	v_1	имеет	<значение>	["является"]	'И'
<ИССловоСоч>	s_2	содержит	<термин>	z_1	'И'

```
<термин>          z1   имеет     <характеристика> x1              'И'
<характеристика>  x1   есть      <падеж>          c1             'И'
<терм-спутникХ>   tx1  имеет     <значение>       ["элементом"]  'И'
<терм-спутникХ>   tx1  имеет     <характеристику> x2             'И'
<характеристика>  x2   есть      <падеж>          c2             'И'
<лексема>         l1   имеет     <характеристику> x3             'И'
<характеристика>  x3   есть      <падеж>          c3             'И'
<падеж>           c1   имеет     <значение>       ["Род"]        'И'
<падеж>           c2   имеет     <значение>       ["Тв"]         'И'
<падеж>           c3   имеет     <значение>       ["Им"]         'И'
<КомпСловоСоч>    k1   имеет     <индекс>         i1             'И'
<СемОтношение>    r1   имеет     <индекс>         (i1+1)         'И'
<лексема>         l1   имеет     <индекс>         (i1+2)
ТО
<термин>          z1   имеет       <тип>        ["Целое"]   'И'
<лексема>         l1   имеет       <тип>        ["Часть"]   'И'
<СемОтношение>    r1   относится к <категория>  ["ЦелоеЧасть"]
```

<*Продукционное правило* 10 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,k1)     ∧ PAggr(Whole,p1,r1)           ∧
PAggr(Whole,p1,l1)      ∧ PAggr(Whole,k1,s1)           ∧
PAggr(Whole,k1,s2)      ∧ PAggr(Whole,s1,tx1)          ∧
PAggr(Whole,r1,v1)      ∧ PHier(Value,v1,"является")   ∧
PAggr(Whole,s2,z1)      ∧ PProp(Character,z1,x1)       ∧
PHier(Category,x1,c1)   ∧ PHier(Value,tx1,"элементом") ∧
PProp(Character,tx1,x2) ∧ PHier(Category,x2,c2)        ∧
PProp(Character,l1,x3)  ∧ PHier(Category,x3,c3)        ∧
PHier(Value,c1,"Род")   ∧ PHier(Value,c2,"Тв")         ∧
PHier(Value,c3,"Им")    ∧ PHier(Index,k1,i1)           ∧
PHier(Index,r1,(i1+1))  ∧ PHier(Index,l1,(i1+2)))      ⊃
(PHier(Category,l1,"Часть") ∧ PHier(Category,z1,"Целое") ∧
PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 11 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>     p1   содержит  <термин>          z1             'И'
<предложение>     p1   содержит  <СемОтношение>    r1             'И'
<предложение>     p1   содержит  <ИССловоСоч>      s1             'И'
<СемОтношение>    r1   содержит  <глагол>          v1             'И'
<глагол>          v1   имеет     <значение>       ["состоит"]    'И'
<СемОтношение>    r1   содержит  <терм-спутникR>   tr1            'И'
<терм-спутникR>   tr1  имеет     <значение>       ["из"]         'И'
<термин>          z1   имеет     <характеристика> x1             'И'
<характеристика>  x1   есть      <падеж>          c1             'И'
<ИССловоСоч>      s1   содержит  <лексема>        l1             'И'
<лексема>         l1   имеет     <характеристика> x2             'И'
<характеристика>  x2   есть      <ЧастьРечи>      c2             'И'
<лексема>         l1   имеет     <характеристика> x3             'И'
<характеристика>  x3   есть      <падеж>          c3             'И'
<падеж>           c1   имеет     <значение>       ["Им"]         'И'
<ЧастьРечи>       c2   имеет     <значение>       ["Сущ"]        'И'
<падеж>           c3   имеет     <значение>       ["Род"]        'И'
<термин>          z1   имеет     <индекс>         i1             'И'
<СемОтношение>    r1   имеет     <индекс>         (i1+1)         'И'
<ИССловоСоч>      s1   имеет     <индекс>         (i1+2)
ТО
<термин>          z1   имеет       <тип>        ["Целое"]   'И'
<лексема>         l1   имеет       <тип>        ["Часть"]   'И'
<СемОтношение>    r1   относится к <категория>  ["ЦелоеЧасть"]
```

<*Продукционное правило* 11 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,z1)     ∧ PAggr(Whole,p1,r1)        ∧
PAggr(Whole,p1,s1)      ∧ PAggr(Whole,s1,l1)        ∧
PAggr(Whole,r1,v1)      ∧ PHier(Value,v1,"состоит") ∧
PAggr(Whole,r1,tr1)     ∧ PHier(Value,tr1,"из")     ∧
PProp(Character,z1,x1)  ∧ PHier(Category,x1,c1)     ∧
PProp(Character,l1,x2)  ∧ PHier(Category,x2,c2)     ∧
PProp(Character,l1,x3)  ∧ PHier(Category,x3,c3)     ∧
PHier(Value,c1,"Им")    ∧ PHier(Value,c2,"Сущ")     ∧
PHier(Value,c3,"Род")   ∧ PHier(Index,z1,i1)        ∧
PHier(Index,r1,(i1+1))  ∧ PHier(Index,s1,(i1+2)))   ⊃
(PHier(Category,l1,"Часть") ∧ PHier(Category,z1,"Целое") ∧
PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 12 на ограниченном подмножестве естественного языка>

ЕСЛИ					
<предложение>	p_1	содержит	<термин>	z_1	'И'
<предложение>	p_1	содержит	<СемОтношение>	r_1	'И'
<предложение>	p_1	содержит	<ИССловоСоч>	s_1	'И'
<СемОтношение>	r_1	содержит	<глагол>	v_1	'И'
<глагол>	v_1	имеет	<значение>	["состоит"]	'И'
<СемОтношение>	r_1	содержит	<терм-спутникR>	tr_1	'И'
<терм-спутникR>	tr_1	имеет	<значение>	["из"]	'И'
<термин>	z_1	имеет	<характеристика>	x_1	'И'
<характеристика>	x_1	есть	<падеж>	c_1	'И'
<ИССловоСоч>	s_1	содержит	<лексема>	l_1	'И'
<лексема>	l_1	имеет	<характеристика>	x_2	'И'
<характеристика>	x_2	есть	<ЧастьРечи>	c_2	'И'
<лексема>	l_1	имеет	<характеристика>	x_3	'И'
<характеристика>	x_3	есть	<падеж>	c_3	'И'
<падеж>	c_1	имеет	<значение>	["Им"]	'И'
<ЧастьРечи>	c_2	имеет	<значение>	["Сущ"]	'И'
<падеж>	c_3	имеет	<значение>	["Род"]	'И'
<термин>	z_1	имеет	<индекс>	i_1	'И'
<СемОтношение>	r_1	имеет	<индекс>	j_1	'И'
<ИССловоСоч>	s_1	имеет	<индекс>	(j_1+1)	
ТО					
<термин>	z_1	имеет	<тип>	["Целое"]	'И'
<лексема>	l_1	имеет	<тип>	["Часть"]	'И'
<СемОтношение>	r_1	относится к	<категория>	["ЦелоеЧасть"]	

<*Продукционное правило* 12 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,z1)     ∧ PAggr(Whole,p1,r1)        ∧
PAggr(Whole,p1,s1)      ∧ PAggr(Whole,s1,l1)        ∧
PAggr(Whole,r1,v1)      ∧ PHier(Value,v1,"состоит") ∧
PAggr(Whole,r1,tr1)     ∧ PHier(Value,tr1,"из")     ∧
PProp(Character,z1,x1)  ∧ PHier(Category,x1,c1)     ∧
PProp(Character,l1,x2)  ∧ PHier(Category,x2,c2)     ∧
PProp(Character,l1,x3)  ∧ PHier(Category,x3,c3)     ∧
PHier(Value,c1,"Им")    ∧ PHier(Value,c2,"Сущ")     ∧
PHier(Value,c3,"Род")   ∧ PHier(Index,z1,i1)        ∧
PHier(Index,r1,j1)      ∧ PHier(Index,s1,(j1+1)))   ⊃
(PHier(Category,l1,"Часть") ∧ PHier(Category,z1,"Целое") ∧
PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 13 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит <ИССловоСоч>     s1           'И'
<предложение>    p1   содержит <СемОтношение>   r1           'И'
<предложение>    p1   содержит <лексема>        l1           'И'
<ИССловоСоч>     s1   содержит <термин>         z1           'И'
<СемОтношение>   r1   содержит <глагол>         v1           'И'
<глагол>         v1   имеет    <значение>       ["входит"]   'И'
<СемОтношение>   r1   содержит <терм-спутникR>  tr1          'И'
<терм-спутникR>  tr1  имеет    <значение>       ["в нее"]    'И'
<термин>         z1   имеет    <характеристика> x1           'И'
<характеристика> x1   есть     <падеж>          c1           'И'
<лексема>        l1   имеет    <характеристика> x2           'И'
<характеристика> x2   есть     <ЧастьРечи>      c2           'И'
<лексема>        l1   имеет    <характеристика> x3           'И'
<характеристика> x3   есть     <падеж>          c3           'И'
<падеж>          c1   имеет    <значение>       ["Им"]       'И'
<ЧастьРечи>      c2   имеет    <значение>       ["Сущ"]      'И'
<падеж>          c3   имеет    <значение>       ["Им"]       'И'
<ИССловоСоч>     s1   имеет    <индекс>         i1           'И'
<СемОтношение>   r1   имеет    <индекс>         j1           'И'
<лексема>        l1   имеет    <индекс>         (j1+1)
ТО
<термин>         z1   имеет       <тип>       ["Целое"]   'И'
<лексема>        l1   имеет       <тип>       ["Часть"]   'И'
<СемОтношение>   r1   относится к <категория> ["ЦелоеЧасть"]
```

<*Продукционное правило* 13 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,s1)     ∧ PAggr(Whole,p1,r1)        ∧
PAggr(Whole,p1,l1)      ∧ PAggr(Whole,s1,z1)        ∧
PAggr(Whole,r1,v1)      ∧ PHier(Value,v1,"входит")  ∧
PAggr(Whole,r1,tr1)     ∧ PHier(Value,tr1,"в нее")  ∧
PProp(Character,z1,x1)  ∧ PHier(Category,x1,c1)     ∧
PProp(Character,l1,x2)  ∧ PHier(Category,x2,c2)     ∧
PProp(Character,l1,x3)  ∧ PHier(Category,x3,c3)     ∧
PHier(Value,c1,"Им")    ∧ PHier(Value,c2,"Сущ")     ∧
PHier(Value,c3,"Им")    ∧ PHier(Index,s1,i1)        ∧
PHier(Index,r1,j1)      ∧ PHier(Index,l1,(j1+1)))   ⊃
(PHier(Category,l1,"Часть") ∧ PHier(Category,z1,"Целое") ∧
PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 14 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит <термин>         z1           'И'
<предложение>    p1   содержит <СемОтношение>   r1           'И'
<предложение>    p1   содержит <список>         q1           'И'
<список>         q1   содержит <ЭлементСписка>  e1           'И'
<СемОтношение>   r1   содержит <глагол>         v1           'И'
<глагол>         v1   имеет    <значение>       ["называют"] 'И'
<СемОтношение>   r1   содержит <терм-спутникR>  tr1          'И'
<терм-спутникR>  tr1  имеет    <значение>       ["комплекс"] 'И'
<термин>         z1   имеет    <характеристика> x1           'И'
<характеристика> x1   есть     <падеж>          c1           'И'
<ЭлементСписка>  e1   имеет    <характеристика> x2           'И'
<характеристика> x2   есть     <ЧастьРечи>      c2           'И'
<ЭлементСписка>  e1   имеет    <характеристика> x3           'И'
<характеристика> x3   есть     <падеж>          c3           'И'
<падеж>          c1   имеет    <значение>       ["Тв"]       'И'
<ЧастьРечи>      c2   имеет    <значение>       ["Сущ"]      'И'
```

```
<падеж>          c3   имеет     <значение>       ["Род"]      'И'
<термин>         z1   имеет     <индекс>         i1           'И'
<СемОтношение>   r1   имеет     <индекс>         (i1+1)       'И'
<список>         q1   имеет     <индекс>         (i1+2)
ТО
<термин>         z1   имеет       <тип>       ["Целое"]   'И'
<список>         q1   имеет       <тип>       ["Части"]   'И'
<СемОтношение>   r1   относится к <категория> ["ЦелоеЧасть"]
```

<*Продукционное правило* 14 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,z1)     ∧ PAggr(Whole,p1,r1)          ∧
PAggr(Whole,p1,q1)      ∧ PAggr(Whole,q1,e1)          ∧
PAggr(Whole,r1,v1)      ∧ PHier(Value,v1,"называют")  ∧
PAggr(Whole,r1,tr1)     ∧ PHier(Value,tr1,"комплекс") ∧
PProp(Character,z1,x1)  ∧ PHier(Category,x1,c1)       ∧
PProp(Character,e1,x2)  ∧ PHier(Category,x2,c2)       ∧
PProp(Character,e1,x3)  ∧ PHier(Category,x3,c3)       ∧
PHier(Value,c1,"Тв")    ∧ PHier(Value,c2,"Сущ")       ∧
PHier(Value,c3,"Род")   ∧ PHier(Index,z1,i1)          ∧
PHier(Index,r1,(i1+1))  ∧ PHier(Index,q1,(i1+2)))     ⊃
(PHier(Category,q1,"Части") ∧ PHier(Category,z1,"Целое") ∧
PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 15 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <ИССловоСоч>     s1                    'И'
<предложение>    p1   содержит  <СемОтношение>   r1                    'И'
<предложение>    p1   содержит  <ИССловоСоч>     s2                    'И'
<ИССловоСоч>     s1   содержит  <термин>         z1                    'И'
<ИССловоСоч>     s2   содержит  <Терм-спутникY>  ty1                   'И'
<ИССловоСоч>     s2   содержит  <лексема>        l1                    'И'
<СемОтношение    r1   содержит  <глагол>         v1                    'И'
<глагол>         v1   имеет     <значение>       ["представляют"]      'И'
<СемОтношение>   r1   содержит  <терм-спутникR>  tr1                   'И'
<терм-спутникR>  tr1  имеет     <значение>       ["собой"]             'И'
<термин>         z1   имеет     <характеристика> x1                    'И'
<характеристика> x1   есть      <падеж>          c1                    'И'
<Терм-спутникY>  ty1  имеет     <значение>       ["Набор"]             'И'
<лексема>        l1   имеет     <характеристика> x2                    'И'
<характеристика> x2   есть      <падеж>          c2                    'И'
<падеж>          c1   имеет     <значение>       ["Им"]                'И'
<падеж>          c2   имеет     <значение>       ["Род"]               'И'
<ИССловоСоч>     s1   имеет     <индекс>         i1                    'И'
<СемОтношение>   r1   имеет     <индекс>         (i1+1)                'И'
<ИССловоСоч>     s2   имеет     <индекс>         (i1+2)
ТО
<термин>         z1   имеет       <тип>       ["Целое"]   'И'
<лексема>        l1   имеет       <тип>       ["Часть"]   'И'
<СемОтношение>   r1   относится к <категория> ["ЦелоеЧасть"]
```

<*Продукционное правило* 15 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,s1)     ∧ PAggr(Whole,p1,r1)              ∧
PAggr(Whole,p1,s2)      ∧ PAggr(Whole,s1,z1)              ∧
PAggr(Whole,s2,l1)      ∧ PAggr(Whole,s2,ty1)             ∧
PAggr(Whole,r1,v1)      ∧ PHier(Value,v1,"представляют") ∧
PAggr(Whole,r1,tr1)     ∧ PHier(Value,tr1,"собой")       ∧
PProp(Character,z1,x1)  ∧ PHier(Category,x1,c1)          ∧
PProp(Character,l1,x2)  ∧ PHier(Category,x2,c2)          ∧
```

```
PHier(Value,ty1,"набор")∧ PHier(Value,c1,"Им")          ∧
PHier(Value,c2,"Род")   ∧ PHier(Index,s1,i1)            ∧
PHier(Index,r1,(i1+1))  ∧ PHier(Index,s2,(i1+2)))       ⊃
(PHier(Category,l1,"Часть") ∧ PHier(Category,z1,"Целое") ∧
PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 16 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит <ИССловоСоч>     s1            'И'
<предложение>    p1   содержит <СемОтношение>   r1            'И'
<предложение>    p1   содержит <список>         q1            'И'
<ИССловоСоч>     s1   содержит <термин>         z1            'И'
<список>         q1   содержит <ЭлементСписка>  e1            'И'
<СемОтношение>   r1   содержит <глагол>         v1            'И'
<глагол>         v1   имеет    <значение>       ["образуют"]  'И'
<термин>         z1   имеет    <характеристику> x1            'И'
<характеристика> x1   есть     <падеж>          c1            'И'
<ЭлементСписка>  e1   есть     <ИССловоСоч>     s2            'И'
<ИССловоСоч>     s2   имеет    <характеристика> x2            'И'
<характеристика> x2   есть     <ЧастьРечи>      c2            'И'
<ЭлементСписка>  e1   имеет    <характеристика> x3            'И'
<характеристика> x3   есть     <падеж>          c3            'И'
<падеж>          c1   имеет    <значение>       ["Вин"]       'И'
<ЧастьРечи>      c2   имеет    <значение>       ["Сущ"]       'И'
<падеж>          c3   имеет    <значение>       ["Им"]        'И'
<ИССловоСоч>     s1   имеет    <индекс>         i1            'И'
<СемОтношение>   r1   имеет    <индекс>         (i1+1)        'И'
<список>         q1   имеет    <индекс>         (i1+2)
ТО
<ИССловоСоч>     s1   имеет       <тип>       ["Целое"]   'И'
<список>         q1   имеет       <тип>       ["Части"]   'И'
<СемОтношение>   r1   относится к <категория> ["ЦелоеЧасть"]
```

<*Продукционное правило* 16 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,s1)          ∧ PAggr(Whole,p1,r1)      ∧
PAggr(Whole,p1,q1)           ∧ PAggr(Whole,q1,e1)      ∧
PAggr(Whole,s1,z1)           ∧ PAggr(Whole,r1,v1)      ∧
PHier(Value,v1,"образуют")   ∧ PProp(Character,z1,x1)  ∧
PHier(Category,x1,c1)        ∧ PProp(Character,e1,s2)  ∧
PProp(Character,s2,x2)       ∧ PHier(Category,x2,c2)   ∧
PProp(Character,e1,x3)       ∧ PHier(Category,x3,c3)   ∧
PHier(Value,c1,"Вин")        ∧ PHier(Value,c2,"Сущ")   ∧
PHier(Value,c3,"Им")         ∧ PHier(Index,z1,i1)      ∧
PHier(Index,r1,(i1+1))       ∧ PHier(Index,q1,(i1+2))) ⊃
(PHier(Category,q1,"Части") ∧ PHier(Category,s1,"Целое") ∧
PHier(Category,r1,"ЦелоеЧасть"))
```

****

<*Продукционное правило* 17 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит <ИССловоСоч>     s1            'И'
<предложение>    p1   содержит <СемОтношение>   r1            'И'
<предложение>    p1   содержит <признак>        h1            'И'
<предложение>    p1   содержит <список>         q1            'И'
<ИССловоСоч>     s1   содержит <терм-спутникХ>  tx1           'И'
<ИССловоСоч>     s1   содержит <термин>         z1            'И'
<список>         q1   содержит <ЭлементСписка>  e1            'И'
<СемОтношение>   r1   содержит <глагол>         v1            'И'
<глагол>         v1   имеет    <значение>       ["входят"]    'И'
```

<признак>	h_1	имеет	<значение>	[“:”]	‘И’
<терм-спутникХ>	tx_1	имеет	<значение>	[“В”]	‘И’
<термин>	z_1	имеет	<характеристика>	x_1	‘И’
<характеристика>	x_1	есть	<падеж>	c_1	‘И’
<ЭлементСписка>	e_1	есть	<ИССловоСоч>	s_2	‘И’
<ИССловоСоч>	s_2	имеет	<характеристика>	x_2	‘И’
<характеристика>	x_2	есть	<падеж>	c_2	‘И’
<падеж>	c_1	имеет	<значение>	[“Вин”]	‘И’
<падеж>	c_2	имеет	<значение>	[“Им”]	‘И’
<ИССловоСоч>	s_1	имеет	<индекс>	i_1	‘И’
<СемОтношение>	r_1	имеет	<индекс>	i_2	‘И’
<признак>	h_1	имеет	<индекс >	i_3	‘И’
<список>	q_1	имеет	<индекс>	(i_3+1)	

ТО

<ИССловоСоч>	s_1	имеет	<тип>	[“Целое”]	‘И’
<список>	q_1	имеет	<тип>	[“Части”]	‘И’
<СемОтношение>	r_1	относится к	<категория>	[“ЦелоеЧасть”]	

<*Продукционное правило* 17 на языке логики предикатов первого порядка>

```
(PAggr(Whole,p1,s1)        ∧ PAggr(Whole,p1,r1)      ∧
PAggr(Whole,p1,h1)         ∧ PAggr(Whole,p1,q1)      ∧
PAggr(Whole,s1,tx1)        ∧ PAggr(Whole,s1,z1)      ∧
PAggr(Whole,q1,e1)         ∧ PAggr(Whole,r1,v1)      ∧
PHier(Value,v1,“входят”)   ∧ PHier(Value,p1,“:”)     ∧
PHier(Value,tx1,“в”)       ∧ PProp(Character,z1,x1)  ∧
PHier(Category,x1,c1)      ∧ PProp(Character,e1,s2)  ∧
PProp(Character,s2,x2)     ∧ PHier(Category,x2,c2)   ∧
PHier(Value,c1,“Вин”)      ∧ PHier(Value,c2,“Им”)    ∧
PHier(Index,s1,i1)         ∧ PHier(Index,r1,i2)      ∧
PHier(Index,h1,i3)         ∧ PHier(Index,q1,(i3+1))) ⊃
(PHier(Category,q1,“Части”) ∧ PHier(Category,s1,“Целое”) ∧
PHier(Category,r1,“ЦелоеЧасть”))
```

2. Извлечение знаний о семантическом отношении «Род-Вид»

<*Продукционное правило* 1 на ограниченном подмножестве естественного языка>

ЕСЛИ

<предложение>	p_1	содержит	<ИССловоСоч>	s_1	‘И’
<ИССловоСоч>	s_1	содержит	<термин>	z_1	‘И’
<ИССловоСоч>	s_1	содержит	<лексема>	l_1	‘И’
<термин>	z_1	имеет	<характеристика>	x_1	‘И’
<характеристика>	x_1	есть	<падеж>	c_1	‘И’
<лексема>	l_1	имеет	<характеристика>	x_2	‘И’
<характеристика>	x_2	есть	<ЧастьРечи>	c_2	‘И’
<лексема>	l_1	имеет	<характеристика>	x_3	‘И’
<характеристика>	x_3	есть	<падеж>	c_3	‘И’
<падеж>	c_1	эквивалентен	<падеж>	c_3	‘И’
<падеж>	c_1	имеет	<значение>	[“Род”]	‘И’
<ЧастьРечи>	c_2	имеет	<значение>	[“Прил”]	‘И’
<падеж>	c_3	имеет	<значение>	[“Род”]	‘И’
<термин>	z_1	имеет	<индекс>	i_1	‘И’
<лексема>	l_1	имеет	<индекс>	(i_1-1)	

ТО

<термин>	z_1	имеет	<тип>	[“Род”]	‘И’
<ИССловоСоч>	s_1	имеет	<тип>	[“Вид”]	‘И’
<СемОтношение>	r_1	относится к	<категория>	[“РодВид”]	

****

<*Продукционное правило* 2 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит      <ИССловоСоч>     s1          'И'
<ИССловоСоч>     s1   содержит      <термин>         z1          'И'
<ИССловоСоч>     s1   содержит      <лексема>        l1          'И'
<термин>         z1   имеет         <характеристику> x1          'И'
<характеристика> x1   есть          <падеж>          c1          'И'
<лексема>        l1   имеет         <характеристика> x2          'И'
<характеристика> x2   есть          <ЧастьРечи>      c2          'И'
<лексема>        l1   имеет         <характеристика> x3          'И'
<характеристика> x3   есть          <падеж>          c3          'И'
<ЧастьРечи>      c2   имеет         <значение>       ["Прил"]    'И'
<падеж>          c1   эквивалентен  <падеж>          c3          'И'
<термин>         z1   имеет         <индекс>         i1          'И'
<лексема>        l1   имеет         <индекс>         (i1+1)
ТО
<термин>         z1   имеет       <тип>       ["Род"]   'И'
<ИССловоСоч>     s1   имеет       <тип>       ["Вид"]   'И'
<СемОтношение>   r1   относится к <категория> ["РодВид"]
```

****

<*Продукционное правило* 3 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <ИССловоСоч>     s1          'И'
<предложение>    p1   содержит  <СемОтношение>   r1          'И'
<предложение>    p1   содержит  <признак>        h1          'И'
<предложение>    p1   содержит  <список>         q1          'И'
<ИССловоСоч>     s1   содержит  <термин>         z1          'И'
<СемОтношение>   r1   содержит  <глагол>         v1          'И'
<глагол>         v1   имеет     <значение>       ["делят"]   'И'
<СемОтношение>   r1   содержит  <Терм-спутникR>  tr1         'И'
<Терм-спутникR>  tr1  имеет     <значение>       ["на"]      'И'
<список>         q1   содержит  <ЭлементСписка>  e2          'И'
<ЭлементСписка>  e2   есть      <ИССловоСоч>     s2          'И'
<ИССловоСоч>     s2   имеет     <характеристика> x1          'И'
<характеристика> x1   есть      <падеж>          c1          'И'
<падеж>          c1   имеет     <значение>       ["Им"]      'И'
<признак>        h1   имеет     <значение>       ["-"]       'И'
<термин>         z1   имеет     <индекс>         i1          'И'
<СемОтношение>   r1   имеет     <индекс>         i2          'И'
<признак>        h1   имеет     <индекс>         i3          'И'
<список>         q1   имеет     <индекс>         (i3+1)
ТО
<ИССловоСоч>     s1   имеет       <тип>     ["Род"]   'И'
<список>         q1   имеет       <тип>     ["Виды"]  'И'
<СемОтношение>   r1   относится к <категория>      ["РодВид"]
```

****

<*Продукционное правило* 4 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит    <ИССловоСоч>     s1       'И'
<предложение>    p1   содержит    <признак>        h1       'И'
<ИССловоСоч>     s1   содержит    <термин>         z1       'И'
<ИССловоСоч>     s1   содержит    <лексема>        l1       'И'
<термин>         z1   имеет       <характеристику> x1       'И'
<характеристика> x1   есть        <падеж>          c1       'И'
<лексема>        l1   имеет       <характеристика> x2       'И'
<характеристика> x2   есть        <ЧастьРечи>      c2       'И'
<лексема>        l1   имеет       <характеристика> x3       'И'
<характеристика> x3   есть        <падеж>          c3       'И'
<ЧастьРечи>      c2   имеет       <значение>       ["Сущ"] 'И'
```

```
<признак>        h1   имеет          <значение>     ['-']   'И'
<падеж>          c1   эквивалентен   <падеж>        c3      'И'
<термин>         z1   имеет          <индекс>       i1      'И'
<лексема>        l1   имеет          <индекс>       (i1+2)  'И'
<признак>        h1   имеет          <индекс>       (i1+1)
ТО
<термин>         z1   имеет       <тип>       ["Род"]   'И'
<ИССловоСоч>     s1   имеет       <тип>       ["Вид"]   'И'
<СемОтношение>   r1   относится к <категория> ["РодВид"]
```

****

<*Продукционное правило* 5 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит      <ИССловоСоч>     s1        'И'
<предложение>    p1   содержит      <признак>        h1        'И'
<ИССловоСоч>     s1   содержит      <термин>         z1        'И'
<ИССловоСоч>     s1   содержит      <лексема>        l1        'И'
<термин>         z1   имеет         <характеристику> x1        'И'
<характеристика> x1   есть          <падеж>          c1        'И'
<лексема>        l1   имеет         <характеристика> x2        'И'
<характеристика> x2   есть          <ЧастьРечи>      c2        'И'
<лексема>        l1   имеет         <характеристика> x3        'И'
<характеристика> x3   есть          <падеж>          c3        'И'
<ЧастьРечи>      c2   имеет         <значение>       ["Сущ"]   'И'
<падеж>          c1   эквивалентен  <падеж>          c3        'И'
<признак>        h1   имеет         <значение>       ['-']     'И'
<термин>         z1   имеет         <индекс>         i1        'И'
<лексема>        l1   имеет         <индекс>         (i1-2)    'И'
<признак>        h1   имеет         <индекс>         (i1-1)
ТО
<термин>         z1   имеет       <тип>       ["Род"]   'И'
<ИССловоСоч>     s1   имеет       <тип>       ["Вид"]   'И'
<СемОтношение>   r1   относится к <категория> ["РодВид"]
```

****

<*Продукционное правило* 6 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <термин>         z1        'И'
<предложение>    p1   содержит  <список>         q1        'И'
<список>         q1   содержит  <ЭлементСписка>  e1        'И'
<ЭлементСписка>  e1   имеет     <характеристика> x1        'И'
<характеристика> x1   есть      <ЧастьРечи>      c1        'И'
<ЧастьРечи>      c1   имеет     <значение>       ["Сущ"]   'И'
<термин>         z1   имеет     <индекс>         i1        'И'
<список>         q1   имеет     <индекс>         (i1+1)
ТО
<термин>         z1   имеет       <тип>       ["Род"]   'И'
<список>         q1   имеет       <тип>       ["Виды"]  'И'
<СемОтношение>   r1   относится к <категория> ["РодВид"]
```

****

<*Продукционное правило* 7 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <термин>         z1      'И'
<предложение>    p1   содержит  <список>         q1      'И'
<список>         q1   содержит  <ЭлементСписка>  e1      'И'
<ЭлементСписка>  e1   есть      <ИССловоСоч>     s1      'И'
<ИССловоСоч>     s1   имеет     <характеристика> x1      'И'
<характеристика> x1   есть      <падеж>          c1      'И'
<падеж>          c1   имеет     <значение>       ["Род"] 'И'
<термин>         z1   имеет     <индекс>         i1      'И'
```

```
<список>          q1   имеет     <индекс>          (i1+1)
ТО
<термин>          z1   имеет        <тип>        ["Род"]   'И'
<список>          q1   имеет        <тип>        ["Виды"]  'И'
<СемОтношение>    r1   относится к <категория>  ["РодВид"]
```

****

<*Продукционное правило* 8 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>     p1   содержит  <СемОтношение>   r1              'И'
<предложение>     p1   содержит  <список>         q1              'И'
<СемОтношение>    r1   содержит  <глагол>         v1              'И'
<глагол>          v1   имеет     <значение>       ["выделяются"] 'И'
<список>          q1   содержит  <ЭлементСписка>  e1              'И'
<ЭлементСписка>   e1   есть      <ИССловоСоч>     s1              'И'
<ИССловоСоч>      s1   содержит  <термин>         z1              'И'
<термин>          z1   имеет     <характеристика> x1              'И'
<характеристика>  x1   есть      <падеж>          c1              'И'
<падеж>           c1   имеет     <значение>       ["Им"]          'И'
<СемОтношение>    r1   имеет     <индекс>         i1              'И'
<список>          q1   имеет     <индекс>         (i1+1)
ТО
<термин>          z1   имеет        <тип>        ["Род"]   'И'
<список>          q1   имеет        <тип>        ["Виды"]  'И'
<СемОтношение>    r1   относится к <категория>  ["РодВид"]
```

****

<*Продукционное правило* 9 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>     p1   содержит  <термин>         z1            'И'
<предложение>     p1   содержит  <СемОтношение>   r1            'И'
<предложение>     p1   содержит  <список>         q1            'И'
<СемОтношение>    r1   содержит  <глагол>         v1            'И'
<глагол>          v1   имеет     <значение>       ["бывают"]   'И'
<список>          q1   содержит  <ЭлементСписка>  e1            'И'
<ЭлементСписка>   e1   имеет     <характеристика> x1            'И'
<характеристика>  x1   есть      <падеж>          c1            'И'
<падеж>           c1   имеет     <значение>       ["Им"]        'И'
<термин>          z1   имеет     <индекс>         i1            'И'
<СемОтношение>    r1   имеет     <индекс>         (i1+1)        'И'
<список>          q1   имеет     <индекс>         (i1+2)
ТО
<термин>          z1   имеет  <тип>            ["Род"]   'И'
<список>          l1   имеет  <тип>            ["Виды"]  'И'
<СемОтношение>    r1   относится к <категория> ["РодВид"]
```

****

<*Продукционное правило* 10 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>     p1   содержит  <СемОтношение>   r1              'И'
<предложение>     p1   содержит  <термин>         z1              'И'
<предложение>     p1   содержит  <признак>        h1              'И'
<предложение>     p1   содержит  <список>         q1              'И'
<СемОтношение>    r1   содержит  <глагол>         v1              'И'
<глагол>          v1   имеет     <значение>       ["выделяются"] 'И'
<термин>          z1   имеет     <характеристика> x1              'И'
<характеристика>  x1   есть      <падеж>          c1              'И'
<признак>         h1   имеет     <значение>       [":"]           'И'
<список>          q1   содержит  <ЭлементСписка>  q1              'И'
<ЭлементСписка>   q1   есть      <ИССловоСоч>     s1              'И'
<ИССловоСоч>      s1   имеет     <характеристика> x2              'И'
```

```
<характеристика> x2   есть      <падеж>          c2              'И'
<падеж>          c1   имеет     <значение>       ["Род"]         'И'
<падеж>          c2   имеет     <значение>       ["Им"]          'И'
<СемОтношение>   r1   имеет     <индекс>         i1              'И'
<термин>         z1   имеет     <индекс>         i2              'И'
<признак>        h1   имеет     <индекс>         (i2+1)          'И'
<список>         q1   имеет     <индекс>         (i2+2)
ТО
<термин>         z1   имеет       <тип>       ["Род"]   'И'
<список>         q1   имеет       <тип>       ["Виды"]  'И'
<СемОтношение>   r1   относится к <категория> ["РодВид"]
```

****

<*Продукционное правило* 11 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <термин>         z1      'И'
<предложение>    p1   содержит  <список>         q1      'И'
<список>         q1   содержит  <ЭлементСписка>  e1      'И'
<ЭлементСписка>  e1   есть      <ИССловоСоч>     s1      'И'
<ИССловоСоч>     s1   имеет     <характеристика> x1      'И'
<характеристика> x1   есть      <падеж>          c1      'И'
<падеж>          c1   имеет     <значение>       ["Им"]  'И'
<термин>         z1   имеет     <индекс>         i1      'И'
<список>         q1   имеет     <индекс>         (i1+1)
ТО
<термин>         z1   имеет       <тип>       ["Род"]   'И'
<список>         q1   имеет       <тип>       ["Виды"]  'И'
<СемОтношение>   r1   относится к <категория> ["РодВид"]
```

****

<*Продукционное правило* 12 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <термин>         z1              'И'
<предложение>    p1   содержит  <СемОтношение>   r1              'И'
<предложение>    p1   содержит  <список>         q1              'И'
<СемОтношение>   r1   содержит  <глагол>         v1              'И'
<глагол>         v1   имеет     <значение>       ["может быть"] 'И'
<список>         q1   содержит  <ЭлементСписка>  e1              'И'
<ЭлементСписка>  e1   есть      <ИССловоСоч>     s1              'И'
<ИССловоСоч>     s1   имеет     <характеристика> x1              'И'
<характеристика> x1   есть      <падеж>          c1              'И'
<падеж>          c1   имеет     <значение>       ["Им"]          'И'
<термин>         z1   имеет     <индекс>         i1              'И'
<СемОтношение>   r1   имеет     <индекс>         (i1+1)          'И'
<список>         q1   имеет     <индекс>         (i1+2)
ТО
<термин>         z1   имеет       <тип>       ["Род"]   'И'
<список>         q1   имеет       <тип>       ["Виды"]  'И'
<СемОтношение>   r1   относится к <категория> ["РодВид"]
```

****

<*Продукционное правило* 13 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<предложение>    p1   содержит  <КомпСловоСоч>   k1    'И'
<предложение>    p1   содержит  <ИССловоСоч>     s1    'И'
<предложение>    p1   содержит  <признак>        h1    'И'
<КомпСловоСоч>   k1   содержит  <ИССловоСоч>     s2    'И'
<ИССловоСоч>     s2   содержит  <термин>         z1    'И'
<ИССловоСоч>     s1   содержит  <термин>         z1    'И'
<термин>         z1   имеет     <характеристика> x1    'И'
<характеристика> x1   есть      <падеж>          c1    'И'
```

<признак> h_1 имеет <значение> ['–'] 'И'
<падеж> c_1 имеет <значение> ["Род"] 'И'
<КомпСловоСоч> k_1 имеет <индекс> i_1 'И'
<признак> h_1 имеет <индекс> (i_1+1) 'И'
<ИССловоСоч> s_1 имеет <индекс> (i_1+2)
ТО
<термин> z_1 имеет <тип> ["Род"] 'И'
<ИССловоСоч> s_1 имеет <тип> ["Вид"] 'И'
<ИССловоСоч> s_2 имеет <тип> ["Вид"] 'И'
<СемОтношение> r_1 относится к <категория> ["РодВид"]

****

<*Продукционное правило* 14 на ограниченном подмножестве естественного языка>

ЕСЛИ
<предложение> p_1 содержит <ИССловоСоч> s_1 'И'
<предложение> p_1 содержит <СемОтношение> r_1 'И'
<предложение> p_1 содержит <ИССловоСоч> s_2 'И'
<ИССловоСоч> s_1 содержит <термин> z_1 'И'
<ИССловоСоч> s_1 содержит <терм-спутникХ> tx_1 'И'
<терм-спутникХ> tx_1 имеет <значение> ["вид"] 'И'
<терм-спутникХ> tx_1 имеет <характеристику> x_1 'И'
<характеристику> x_1 есть <падеж> c_1 'И'
<СемОтношение> r_1 имеет <индекс> (i_1+1) 'И'
<СемОтношение> r_1 содержит <глагол> v_1 'И'
<глагол> v_1 имеет <значение> ["являются"] 'И'
<падеж> c_1 имеет <значение> ["Род"] 'И'
<ИССловоСоч> s_1 имеет <индекс> i_1 'И'
<ИССловоСоч> s_2 имеет <индекс> (i_1+2)
ТО
<термин> z_1 имеет <тип> ["Род"] 'И'
<ИССловоСоч> s_2 имеет <тип> ["Вид"] 'И'
<СемОтношение> r_1 относится к <категория> ["РодВид"]

****

<*Продукционное правило* 15 на ограниченном подмножестве естественного языка>

ЕСЛИ
<предложение> p_1 содержит <ИССловоСоч> s_1 'И'
<предложение> p_1 содержит <СемОтношение> r_1 'И'
<предложение> p_1 содержит <ИССловоСоч> s_2 'И'
<ИССловоСоч> s_1 содержит <термин> z_1 'И'
<ИССловоСоч> s_1 содержит <терм-спутникХ> tx_1 'И'
<терм-спутникХ> tx_1 имеет <значение> ["вид"] 'И'
<терм-спутникХ> tx_1 имеет <характеристику> x_1 'И'
<характеристику> x_1 есть <падеж> c_1 'И'
<СемОтношение> r_1 содержит <глагол> v_1 'И'
<глагол> v_1 имеет <значение> ["являются"] 'И'
<падеж> c_1 имеет <значение> ["Тв"] 'И'
<ИССловоСоч> s_1 имеет <индекс> i_1 'И'
<СемОтношение> r_1 имеет <индекс> (i_1+1) 'И'
<ИССловоСоч> s_2 имеет <индекс> (i_1+2)
ТО
<термин> z_1 имеет <тип> ["Род"] 'И'
<ИССловоСоч> s_2 имеет <тип> ["Вид"] 'И'
<СемОтношение> r_1 относится к <категория> ["РодВид"]

****

<*Продукционное правило* 16 на ограниченном подмножестве естественного языка>

ЕСЛИ
<предложение> p_1 содержит <ИССловоСоч> s_1 'И'
<предложение> p_1 содержит <СемОтношение> r_1 'И'
<предложение> p_1 содержит <ИССловоСоч> s_2 'И'

```
<ИССловоСоч>      s1   содержит  <термин>          z1       'И'
<ИССловоСоч>      s1   содержит  <терм-спутникХ>   tx1      'И'
<терм-спутникХ>   tx1  имеет     <значение>        ["вид"]  'И'
<терм-спутникХ>   tx1  имеет     <характеристику>  x1       'И'
<характеристику>  x1   есть      <падеж>           c1       'И'
<СемОтношение>    r1   содержит  <глагол>          v1       'И'
<глагол>          v1   имеет     <значение>        ['-']    'И'
<падеж>           c1   имеет     <значение>        ["Им"]   'И'
<ИССловоСоч>      s1   имеет     <индекс>          i1       'И'
<СемОтношение>    r1   имеет     <индекс>          (i1+1)   'И'
<ИССловоСоч>      s2   имеет     <индекс>          (i1+2)
ТО
<термин>          z1   имеет       <тип>       ["Род"]   'И'
<ИССловоСоч>      s2   имеет       <тип>       ["Вид"]   'И'
<СемОтношение>    r1   относится к <категория> ["РодВид"]
```

3. Морфологический анализ

3.1. Продукции для определения части речи неизменяемых словоформ

<*Продукционное правило* 1 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>     p1      содержит      <Лексема>         l1       'И'
<Лексема>         l1      принадлежит   <Словарь>         d1       'И'
<Словарь>         d1      содержит      <Вектор>          RWord    'И'
<Вектор>          RWord   содержит      <Компонент>       g1       'И'
<Вектор>          RWord   содержит      <Компонент>       g2       'И'
<Словарь>         d1      имеет         <Характеристика>  x1       'И'
<Характеристика>  x1      есть          <Имя>             n1       'И'
<Компонент>       g1      есть          <ЧастьРечи>       c1       'И'
<Компонент>       g2      есть          <ГотСловоформа>   c2       'И'
<Имя>             n1      имеет         <Значение>        ["СГС"]  'И'
<ЧастьРечи>       c1      имеет         <Значение>        ["сущ"]  'И'
<ГотСловоформа>   c2      эквивалентна  <Лексема>         l1
ТО
<Лексема>         l1      имеет  <Характеристика>  x2   'И'
<Характеристика>  x2      есть   <ЧастьРечи>       c1   'И'
<ЧастьРечи>       c1      имеет  <Значение>        ["сущ"]
```

<*Продукционное правило* 1 на языке логики предикатов первого порядка>

```
(PAggr("Whole",p1, l1)      ∧ PAggr("Part",l1,d1)        ∧
PAggr("Whole",d1, RWord)    ∧ PAggr("Whole",RWord, g1)   ∧
PAggr("Whole",RWord, g2)    ∧ PProp("Character",d1,x1)   ∧
PHier("Category",x1,n1)     ∧ PHier("Category",g1,c1)    ∧
PHier("Category",g2,c2)     ∧ PHier("Value",n1,"СГС")    ∧
PHier("Value",c1,"сущ")     ∧ PEquiv("Identity",c2,l1))⊃
(PProp("Character",l1,x2)   ∧ PHier("Category",x2,c1) ∧
(PHier("Value",c1,"сущ")    )
```

****

<*Продукционное правило* 2 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>     p1      содержит      <Лексема>         l1       'И'
<Лексема>         l1      принадлежит   <Словарь>         d1       'И'
<Словарь>         d1      содержит      <Вектор>          RWord    'И'
<Вектор>          RWord   содержит      <Компонент>       g1       'И'
<Вектор>          RWord   содержит      <Компонент>       g2       'И'
<Словарь>         d1      имеет         <Характеристика>  x1       'И'
<Характеристика>  x1      есть          <Имя>             n1       'И'
<Компонент>       g1      есть          <ЧастьРечи>       c1       'И'
<Компонент>       g2      есть          <ГотСловоформа>   c2       'И'
```

<Имя> n_1 имеет <Значение> ["СГС"] 'И'
<ЧастьРечи> c_1 имеет <Значение> ["наречие"]
ТО
<Лексема> l_1 имеет <Характеристика> x_2 'И'
<Характеристика> x_2 есть <ЧастьРечи> c_1 'И'
<ЧастьРечи> c_1 имеет <Значение> ["наречие"]

<*Продукционное правило* 2 на языке логики предикатов первого порядка>

(PAggr("Whole",p_1, l_1) ∧ PAggr("Part",l_1,d_1) ∧
PAggr("Whole",d_1, *RWord*) ∧ PAggr("Whole",*RWord*, g_1) ∧
PAggr("Whole",*RWord*, g_2) ∧ PProp("Character",d_1,x_1) ∧
PHier("Category",x_1,n_1) ∧ PHier("Category",g_1,c_1) ∧
PHier("Category",g_2,c_2) ∧ PHier("Value",n_1,"СГС") ∧
PHier("Value",c_1,"наречие") ∧ PEquiv("Identity",c_2,l_1))⊃
(PProp("Character",l_1,x_2) ∧ PHier("Category",x_2,c_1) ∧
PHier("Value",c_1,"наречие"))

****

<*Продукционное правило* 3 на ограниченном подмножестве естественного языка>

ЕСЛИ
<Предложение> p_1 содержит <Лексема> l_1 'И'
<Лексема> l_1 принадлежит <Словарь> d_1 'И'
<Словарь> d_1 содержит <Вектор> *RWord* 'И'
<Вектор> *RWord* содержит <Компонент> g_1 'И'
<Вектор> *RWord* содержит <Компонент> g_2 'И'
<Словарь> d_1 имеет <Характеристика> x_1 'И'
<Характеристика> x_1 есть <Имя> n_1 'И'
<Компонент> g_1 есть <ЧастьРечи> c_1 'И'
<Компонент> g_2 есть <ГотСловоформа> c_2 'И'
<Имя> n_1 имеет <Значение> ["СГС"] 'И'
<ЧастьРечи> c_1 имеет <Значение> ["местоимение"] 'И'
<ГотСловоформа> c_2 эквивалентна <Лексема> l_1
ТО
<Лексема> l_1 имеет <Характеристика> x_2 'И'
<Характеристика> x_2 есть <ЧастьРечи> c_1 'И'
<ЧастьРечи> c_1 имеет <Значение> ["местоимение"]

<*Продукционное правило* 3 на языке логики предикатов первого порядка>

(PAggr("Whole",p_1, l_1) ∧ PAggr("Part",l_1,d_1) ∧
PAggr("Whole",d_1, *RWord*) ∧ PAggr("Whole",*RWord*, g_1) ∧
PAggr("Whole",*RWord*, g_2) ∧ PProp("Character",d_1,x_1) ∧
PHier("Category",x_1,n_1) ∧ PHier("Category",g_1,c_1) ∧
PHier("Category",g_2,c_2) ∧ PHier("Value",n_1,"СГС") ∧
PHier("Value",c_1,"местоимение") ∧ PEquiv("Identity",c_2,l_1))⊃
(PProp("Character",l_1,x_2) ∧ PHier("Category",x_2,c_1) ∧
PHier("Value",c_1,"местоимение"))

****

<*Продукционное правило* 4 на ограниченном подмножестве естественного языка>

ЕСЛИ
<Предложение> p_1 содержит <Лексема> l_1 'И'
<Лексема> l_1 принадлежит <Словарь> d_1 'И'
<Словарь> d_1 содержит <Вектор> *RWord* 'И'
<Вектор> *RWord* содержит <Компонент> g_1 'И'
<Вектор> *RWord* содержит <Компонент> g_2 'И'
<Словарь> d_1 имеет <Характеристика> x_1 'И'
<Характеристика> x_1 есть <Имя> n_1 'И'
<Компонент> g_1 есть <ЧастьРечи> c_1 'И'
<Компонент> g_2 есть <ГотСловоформа> c_2 'И'
<Имя> n_1 имеет <Значение> ["СГС"] 'И'

```
<ЧастьРечи>      c1      имеет          <Значение>       ["числительное"] 'И'
<ГотСловоформа>  c2      эквивалентна   <Лексема>        l1
ТО
<Лексема>        l1      имеет   <Характеристика> x2   'И'
<Характеристика> x2      есть    <ЧастьРечи>      c1   'И'
<ЧастьРечи>      c1      имеет   <Значение>       ["числительное"]
```

<*Продукционное правило* 4 на языке логики предикатов первого порядка>

```
(PAggr("Whole",p1, l1)        ∧ PAggr("Part",l1,d1)        ∧
PAggr("Whole",d1, RWord)      ∧ PAggr("Whole",RWord, g1)   ∧
PAggr("Whole",RWord, g2)      ∧ PProp("Character",d1,x1)   ∧
PHier("Category",x1,n1)       ∧ PHier("Category",g1,c1)    ∧
PHier("Category",g2,c2)       ∧ PHier("Value",n1,"СГС")    ∧
PHier("Value",c1,"числительное") ∧ PEquiv("Identity",c2,l1))⊃
(PProp("Character",l1,x2)         ∧ PHier("Category",x2,c1) ∧
PHier("Value",c1,"числительное"))
```

****

<*Продукционное правило* 5 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1      содержит       <Лексема>        l1      'И'
<Лексема>        l1      принадлежит    <Словарь>        d1      'И'
<Словарь>        d1      содержит       <Вектор>         RWord   'И'
<Вектор>         RWord   содержит       <Компонент>      g1      'И'
<Вектор>         RWord   содержит       <Компонент>      g2      'И'
<Словарь>        d1      имеет          <Характеристика> x1      'И'
<Характеристика> x1      есть           <Имя>            n1      'И'
<Компонент>      g1      есть           <ЧастьРечи>      c1      'И'
<Компонент>      g2      есть           <ГотСловоформа>  c2      'И'
<Имя>            n1      имеет          <Значение>       ["СГС"] 'И'
<ЧастьРечи>      c1      имеет          <Значение>       ["предлог"] 'И'
<ГотСловоформа>  c2      эквивалентна   <Лексема>        l1
ТО
<Лексема>        l1      имеет   <Характеристика> x2   'И'
<Характеристика> x2      есть    <ЧастьРечи>      c1   'И'
<ЧастьРечи>      c1      имеет   <Значение>       ["предлог"]
```

<*Продукционное правило* 5 на языке логики предикатов первого порядка>

```
(PAggr("Whole",p1, l1)        ∧ PAggr("Part",l1,d1)        ∧
PAggr("Whole",d1, RWord)      ∧ PAggr("Whole",RWord, g1)   ∧
PAggr("Whole",RWord, g2)      ∧ PProp("Character",d1,x1)   ∧
PHier("Category",x1,n1)       ∧ PHier("Category",g1,c1)    ∧
PHier("Category",g2,c2)       ∧ PHier("Value",n1,"СГС")    ∧
PHier("Value",c1,"предлог")   ∧ PEquiv("Identity",c2,l1))⊃
(PProp("Character",l1,x2)     ∧ PHier("Category",x2,c1) ∧
PHier("Value",c1,"предлог"))
```

****

<*Продукционное правило* 6 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1      содержит       <Лексема>        l1      'И'
<Лексема>        l1      принадлежит    <Словарь>        d1      'И'
<Словарь>        d1      содержит       <Вектор>         RWord   'И'
<Вектор>         RWord   содержит       <Компонент>      g1      'И'
<Вектор>         RWord   содержит       <Компонент>      g2      'И'
<Словарь>        d1      имеет          <Характеристика> x1      'И'
<Характеристика> x1      есть           <Имя>            n1      'И'
<Компонент>      g1      есть           <ЧастьРечи>      c1      'И'
<Компонент>      g2      есть           <ГотСловоформа>  c2      'И'
<ЧастьРечи>      c1      имеет          <Значение>       ["союз"] 'И'
```

```
<ГотСловоформа>  c2      эквивалентна  <Лексема>        l1
ТО
<Лексема>        l1      имеет  <Характеристика> x2   'И'
<Характеристика> x2      есть   <ЧастьРечи>      c1   'И'
<ЧастьРечи>      c1      имеет  <Значение>       ["союз"]
```

<*Продукционное правило* 6 на языке логики предикатов первого порядка>

```
(PAggr("Whole",p1, l1)      ∧ PAggr("Part",l1,d1)        ∧
PAggr("Whole",d1, RWord)    ∧ PAggr("Whole",RWord, g1)   ∧
PAggr("Whole",RWord, g2)    ∧ PProp("Character",d1,x1)   ∧
PHier("Category",x1,n1)     ∧ PHier("Category",g1,c1)    ∧
PHier("Category",g2,c2)     ∧ PHier("Value",n1,"СГС")    ∧
PHier("Value",c1,"союз")    ∧ PEquiv("Identity",c2,l1))⊃
(PProp("Character",l1,x2)   ∧ PHier("Category",x2,c1) ∧
PHier("Value",c1,"союз"))
```

****

<*Продукционное правило* 7 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1      содержит      <Лексема>        l1      'И'
<Лексема>        l1      принадлежит   <Словарь>        d1      'И'
<Словарь>        d1      содержит      <Вектор>         RWord   'И'
<Вектор>         RWord   содержит      <Компонент>      g1      'И'
<Вектор>         RWord   содержит      <Компонент>      g2      'И'
<Словарь>        d1      имеет         <Характеристика> x1      'И'
<Характеристика> x1      есть          <Имя>            n1      'И'
<Компонент>      g1      есть          <ЧастьРечи>      c1      'И'
<Компонент>      g2      есть          <ГотСловоформа>  c2      'И'
<Имя>            n1      имеет         <Значение>       ["СГС"] 'И'
<ЧастьРечи>      c1      имеет         <Значение>       ["вводное слово"]'И'
<ГотСловоформа>  c2      эквивалентна  <Лексема>        l1
ТО
<Лексема>        l1      имеет  <Характеристика> x2   'И'
<Характеристика> x2      есть   <ЧастьРечи>      c1   'И'
<ЧастьРечи>      c1      имеет  <Значение>       ["вводное слово"]
```

<*Продукционное правило* 7 на языке логики предикатов первого порядка>

```
(PAggr("Whole",p1, l1)       ∧ PAggr("Part",l1,d1)        ∧
PAggr("Whole",d1, RWord)     ∧ PAggr("Whole",RWord, g1)   ∧
PAggr("Whole",RWord, g2)     ∧ PProp("Character",d1,x1)   ∧
PHier("Category",x1,n1)      ∧ PHier("Category",g1,c1)    ∧
PHier("Category",g2,c2)      ∧ PHier("Value",n1,"СГС")    ∧
PHier("Value",c1,"вводное слово") ∧ PEquiv("Identity",c2,l1))⊃
(PProp("Character",l1,x2)          ∧ PHier("Category",x2,c1) ∧
PHier("Value",c1,"вводное слово"))
```

****

<*Продукционное правило* 8 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1      содержит      <Лексема>        l1      'И'
<Лексема>        l1      принадлежит   <Словарь>        d1      'И'
<Словарь>        d1      содержит      <Вектор>         RWord   'И'
<Вектор>         RWord   содержит      <Компонент>      g1      'И'
<Вектор>         RWord   содержит      <Компонент>      g2      'И'
<Словарь>        d1      имеет         <Характеристика> x1      'И'
<Характеристика> x1      есть          <Имя>            n1      'И'
<Компонент>      g1      есть          <ЧастьРечи>      c1      'И'
<Компонент>      g2      есть          <ГотСловоформа>  c2      'И'
<Имя>            n1      имеет         <Значение>       ["СГС"] 'И'
<ЧастьРечи>      c1      имеет         <Значение>       ["деепричастие"] 'И'
```

```
<ГотСловоформа>  c2      эквивалентна  <Лексема>        l1
ТО
<Лексема>        l1      имеет  <Характеристика> x2   'И'
<Характеристика> x2      есть   <ЧастьРечи>      c1   'И'
<ЧастьРечи>      c1      имеет  <Значение>       ["деепричастие"]
```

<*Продукционное правило* 8 на языке логики предикатов первого порядка>

```
(PAggr("Whole",p1, l1)          ∧ PAggr("Part",l1,d1)          ∧
PAggr("Whole",d1, RWord)        ∧ PAggr("Whole",RWord, g1)     ∧
PAggr("Whole",RWord, g2)        ∧ PProp("Character",d1,x1)     ∧
PHier("Category",x1,n1)         ∧ PHier("Category",g1,c1)      ∧
PHier("Category",g2,c2)         ∧ PHier("Value",n1,"СГС")      ∧
PHier("Value",c1,"деепричастие") ∧ PEquiv("Identity",c2,l1))⊃
(PProp("Character",l1,x2)        ∧ PHier("Category",x2,c1) ∧
PHier("Value",c1,"деепричастие"))
```

****

<*Продукционное правило* 9 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1      содержит      <Лексема>        l1      'И'
<Лексема>        l1      принадлежит   <Словарь>        d1      'И'
<Словарь>        d1      содержит      <Вектор>         RWord   'И'
<Вектор>         RWord   содержит      <Компонент>      g1      'И'
<Вектор>         RWord   содержит      <Компонент>      g2      'И'
<Словарь>        d1      имеет         <Характеристика> x1      'И'
<Характеристика> x1      есть          <Имя>            n1      'И'
<Компонент>      g1      есть          <ЧастьРечи>      c1      'И'
<Компонент>      g2      есть          <ГотСловоформа>  c2      'И'
<Имя>            n1      имеет         <Значение>       ["СГС"] 'И'
<ЧастьРечи>      c1      имеет         <Значение>       ["частица"]'И'
<ГотСловоформа>  c2      эквивалентна  <Лексема>        l1
ТО
<Лексема>        l1      имеет  <Характеристика> x2   'И'
<Характеристика> x2      есть   <ЧастьРечи>      c1   'И'
<ЧастьРечи>      c1      имеет  <Значение>       ["частица"]
```

<*Продукционное правило* 9 на языке логики предикатов первого порядка>

```
(PAggr("Whole",p1, l1)          ∧ PAggr("Part",l1,d1)          ∧
PAggr("Whole",d1, RWord)        ∧ PAggr("Whole",RWord, g1)     ∧
PAggr("Whole",RWord, g2)        ∧ PProp("Character",d1,x1)     ∧
PHier("Category",x1,n1)         ∧ PHier("Category",g1,c1)      ∧
PHier("Category",g2,c2)         ∧ PHier("Value",n1,"СГС")      ∧
PHier("Value",c1,"частица")     ∧ PEquiv("Identity",c2,l1))⊃
(PProp("Character",l1,x2)       ∧ PHier("Category",x2,c1) ∧
PHier("Value",c1,"частица"))
```

****

3.2. Продукции для определения части речи изменяемых словоформ

<*Продукционное правило* 10 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1      содержит      <Лексема>        l1      'И'
<Лексема>        l1      содержит      <Основа>         b1      'И'
<Лексема>        l1      содержит      <Окончание>      o1      'И'
<Основа>         b1      принадлежит   <Словарь>        d1      'И'
<Окончание>      o1      принадлежит   <Словарь>        d2      'И'
<Словарь>        d1      содержит      <Вектор>         Basis   'И'
<Словарь>        d2      содержит      <Вектор>         End     'И'
<Вектор>         Basis   содержит      <Компонент>      g1      'И'
<Вектор>         Basis   содержит      <Компонент>      g2      'И'
<Вектор>         End     содержит      <Компонент>      g3      'И'
```

```
<Вектор>          End     содержит      <Компонент>      g4      'И'
<Словарь>         d1      имеет         <Характеристика> x1      'И'
<Словарь>         d2      имеет         <Характеристика> x2      'И'
<Характеристика>  x1      есть          <Имя>            n1      'И'
<Характеристика>  x2      есть          <Имя>            n2      'И'
<Компонент>       g1      есть          <ЧастьРечи>      c1      'И'
<Компонент>       g2      есть          <Основа>         c2      'И'
<Компонент>       g3      есть          <ЧастьРечи>      c3      'И'
<Компонент>       g4      есть          <Окончание>      c4      'И'
<Имя>             n1      имеет         <Значение>    ["Словарь основ"]   'И'
<Имя>             n2      имеет         <Значение>    ["Словарь окончаний"]'И'
<ЧастьРечи>       c1      имеет         <Значение>    ["сущ"]   'И'
<ЧастьРечи>       c3      имеет         <Значение>    ["сущ"]   'И'
<Основа>          c2      эквивалентна  <Основа>         b1      'И'
<Окончание>       c4      эквивалентна  <Окончание>      o1      'И'
<ЧастьРечи>       c1      эквивалентна  <ЧастьРечи>      c3
ТО
<Лексема>         l1      имеет  <Характеристика> x1 'И'
<Характеристика>  x1      есть   <ЧастьРечи>      c1 'И'
<ЧастьРечи>       c1      имеет  <Значение>       ["сущ"]
**********************************************************************************
```

<*Продукционное правило* 11 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>     p1      содержит      <Лексема>        l1      'И'
<Лексема>         l1      содержит      <Основа>         b1      'И'
<Лексема>         l1      содержит      <Окончание>      o1      'И'
<Основа>          b1      принадлежит   <Словарь>        d1      'И'
<Окончание>       o1      принадлежит   <Словарь>        d2      'И'
<Словарь>         d1      содержит      <Вектор>         Basis   'И'
<Словарь>         d2      содержит      <Вектор>         End     'И'
<Вектор>          Basis   содержит      <Компонент>      g1      'И'
<Вектор>          Basis   содержит      <Компонент>      g2      'И'
<Вектор>          End     содержит      <Компонент>      g3      'И'
<Вектор>          End     содержит      <Компонент>      g4      'И'
<Словарь>         d1      имеет         <Характеристика> x1      'И'
<Словарь>         d2      имеет         <Характеристика> x2      'И'
<Характеристика>  x1      есть          <Имя>            n1      'И'
<Характеристика>  x2      есть          <Имя>            n2      'И'
<Компонент>       g1      есть          <ЧастьРечи>      c1      'И'
<Компонент>       g2      есть          <Основа>         c2      'И'
<Компонент>       g3      есть          <ЧастьРечи>      c3      'И'
<Компонент>       g4      есть          <Окончание>      c4      'И'
<Имя>             n1      имеет         <Значение>    ["Словарь основ"]'И'
<Имя>             n2      имеет         <Значение>    ["Словарь окончаний"]'И'
<ЧастьРечи>       c1      имеет         <Значение>    ["прил"]  'И'
<ЧастьРечи>       c3      имеет         <Значение>    ["прил"]  'И'
<Основа>          c2      эквивалентна  <Основа>         b1      'И'
<Окончание>       c4      эквивалентна  <Окончание>      o1      'И'
<ЧастьРечи>       c1      эквивалентна  <ЧастьРечи>      c3
ТО
<Лексема>         l1      имеет  <Характеристика> x1   'И'
<Характеристика>  x1      есть   <ЧастьРечи>      c1   'И'
<ЧастьРечи>       c1      имеет  <Значение>       ["прил"]
**********************************************************************************
```

<*Продукционное правило* 12 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>     p1      содержит      <Лексема>        l1      'И'
<Лексема>         l1      содержит      <Основа>         b1      'И'
<Лексема>         l1      содержит      <Окончание>      o1      'И'
<Основа>          b1      принадлежит   <Словарь>        d1      'И'
```

```
<Окончание>      o1     принадлежит   <Словарь>         d2      'И'
<Словарь>        d1     содержит      <Вектор>          Basis   'И'
<Словарь>        d2     содержит      <Вектор>          End     'И'
<Вектор>         Basis  содержит      <Компонент>       g1      'И'
<Вектор>         Basis  содержит      <Компонент>       g2      'И'
<Вектор>         End    содержит      <Компонент>       g3      'И'
<Вектор>         End    содержит      <Компонент>       g4      'И'
<Словарь>        d1     имеет         <Характеристика>  x1      'И'
<Словарь>        d2     имеет         <Характеристика>  x2      'И'
<Характеристика> x1     есть          <Имя>             n1      'И'
<Характеристика> x2     есть          <Имя>             n2      'И'
<Компонент>      g1     есть          <ЧастьРечи>       c1      'И'
<Компонент>      g2     есть          <Основа>          c2      'И'
<Компонент>      g3     есть          <ЧастьРечи>       c3      'И'
<Компонент>      g4     есть          <Окончание>       c4      'И'
<Имя>            n1     имеет         <Значение>     ["Словарь основ"]'И'
<Имя>            n2     имеет         <Значение>     ["Словарь окончаний"]'И'
<ЧастьРечи>      c1     имеет         <Значение>     ["причастие"]  'И'
<ЧастьРечи>      c3     имеет         <Значение>     ["причастие"]  'И'
<Основа>         c2     эквивалентна  <Основа>          b1      'И'
<Окончание>      c4     эквивалентна  <Окончание>       o1      'И'
<ЧастьРечи>      c1     эквивалентна  <ЧастьРечи>       c3
ТО
<Лексема>        l1     имеет  <Характеристика>  x1    'И'
<Характеристика> x1     есть   <ЧастьРечи>       c1    'И'
<ЧастьРечи>      c1     имеет  <Значение>        ["причастие"]
**********************************************************************************************
```

<*Продукционное правило* 13 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1     содержит      <Лексема>         l1      'И'
<Лексема>        l1     содержит      <Основа>          b1      'И'
<Лексема>        l1     содержит      <Окончание>       o1      'И'
<Основа>         b1     принадлежит   <Словарь>         d1      'И'
<Окончание>      o1     принадлежит   <Словарь>         d2      'И'
<Словарь>        d1     содержит      <Вектор>          Basis   'И'
<Словарь>        d2     содержит      <Вектор>          End     'И'
<Вектор>         Basis  содержит      <Компонент>       g1      'И'
<Вектор>         Basis  содержит      <Компонент>       g2      'И'
<Вектор>         End    содержит      <Компонент>       g3      'И'
<Вектор>         End    содержит      <Компонент>       g4      'И'
<Словарь>        d1     имеет         <Характеристика>  x1      'И'
<Словарь>        d2     имеет         <Характеристика>  x2      'И'
<Характеристика> x1     есть          <Имя>             n1      'И'
<Характеристика> x2     есть          <Имя>             n2      'И'
<Компонент>      g1     есть          <ЧастьРечи>       c1      'И'
<Компонент>      g2     есть          <Основа>          c2      'И'
<Компонент>      g3     есть          <ЧастьРечи>       c3      'И'
<Компонент>      g4     есть          <Окончание>       c4      'И'
<Имя>            n1     имеет         <Значение>     ["Словарь основ"]'И'
<Имя>            n2     имеет         <Значение>     ["Словарь окончаний"]'И'
<ЧастьРечи>      c1     имеет         <Значение>     ["глагол"]'И'
<ЧастьРечи>      c3     имеет         <Значение>     ["глагол"]'И'
<Основа>         c2     эквивалентна  <Основа>          b1      'И'
<Окончание>      c4     эквивалентна  <Окончание>       o1      'И'
<ЧастьРечи>      c1     эквивалентна  <ЧастьРечи>       c3
ТО
<Лексема>        l1     имеет  <Характеристика>  x1    'И'
<Характеристика> x1     есть   <ЧастьРечи>       c1
<ЧастьРечи>      c1     имеет  <Значение>        ["глагол"]
**********************************************************************************************
```

3.3. Продукции для определения номера флективного класса, статической и динамической морфологической информации

<*Продукционное правило* 14 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит      <Лексема>        l1    'И'
<Лексема>        l1   содержит      <Основа>         b1    'И'
<Основа>         b1   принадлежит   <Словарь>        d1    'И'
<Словарь>        d1   содержит      <Вектор>         Fk    'И'
<Вектор>         Fk   содержит      <Компонент>      g1    'И'
<Вектор>         Fk   содержит      <Компонент>      g2    'И'
<Словарь>        d1   имеет         <Характеристика> x1    'И'
<Характеристика> x1   есть          <Имя>            n1    'И'
<Компонент>      g1   есть          <Код ФК>         c1    'И'
<Компонент>      g2   есть          <Основа>         c2    'И'
<Имя>            n1   имеет         <Значение>       ["Словарь ФК"] 'И'
<Основа>         c2   эквивалентна  <Основа>         b1
ТО
<Лексема>        l1   имеет  <Характеристика>   x1   'И'
<Характеристика> x1   есть   <Код ФК>           c1
```

****

<*Продукционное правило* 15 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит      <Лексема>                l1      'И'
<Лексема>        l1   принадлежит   <Словарь>                d1      'И'
<Словарь>        d1   содержит      <Вектор>                 Smi           'И'
<Вектор>         Smi  содержит      <Компонент>              g1      'И'
<Вектор>         Smi  содержит      <Компонент>              g2      'И'
<Вектор>         Smi  содержит      <Компонент>              g3      'И'
<Вектор>         Smi  содержит      <Компонент>              g4      'И'
<Словарь>        d1   имеет         <Характеристика>         x1      'И'
<Лексема>        l1   имеет         <Характеристика>         x2      'И'
<Лексема>        l1   имеет         <Характеристика>         x3      'И'
<Характеристика> x1   есть          <Имя>                    n1      'И'
<Характеристика> x2   есть          <КодФК>                  c5      'И'
<Характеристика> x3   есть          <ЧастьРечи>              c6      'И'
<Компонент>      g1   есть          <Код ФК>                 c1      'И'
<Компонент>      g2   есть          <ЧастьРечи>                c2          'И'
<Компонент>      g3   есть          <Род>                      c3          'И'
<Компонент>      g4   есть          <Признак одушевленности>   c4          'И'
<Имя>            n1   имеет         <Значение>           ["Словарь СМИ"]'И'
<ЧастьРечи>      c6   имеет         <Значение>               ["сущ"]     'И'
<Код ФК>         c1   эквивалентен  <КодФК>                  c5          'И'
<ЧастьРечи>      c2   эквивалентен  <ЧастьРечи>              c6
ТО
<Лексема>        l1   имеет         <Характеристика>         x4          'И'
<Характеристика> x1   есть          <Род>                    c3          'И'
<Лексема>        l1   имеет         <Характеристика>         x5          'И'
<Характеристика> x1   есть          <Признак одушевленности> c4
```

****

<*Продукционное правило* 16 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит      <Лексема>        l1    'И'
<Лексема>        l1   содержит      <Окончание>      o1    'И'
<Окончание>      o1   принадлежит   <Словарь>        d1    'И'
<Словарь>        d1   содержит      <Вектор>         Dmi   'И'
<Вектор>         Dmi  содержит      <Компонент>      g1    'И'
<Вектор>         Dmi  содержит      <Компонент>      g2    'И'
<Вектор>         Dmi  содержит      <Компонент>      g3    'И'
<Вектор>         Dmi  содержит      <Компонент>      g4    'И'
```

<Вектор>	*Dmi*	содержит	<Компонент>	g_5	'И'	
<Словарь>	d_1	имеет	<Характеристика>	x_1	'И'	
<Лексема>	l_1	имеет	<Характеристика>	x_2	'И'	
<Лексема>	l_1	имеет	<Характеристика>	x_3	'И'	
<Характеристика>	x_1	есть	<Имя>	n_1	'И'	
<Характеристика>	x_2	есть	<КодФК>	c_6	'И'	
<Характеристика>	x_3	есть	<ЧастьРечи>	c_7	'И'	
<Компонент>	g_1	есть	<Код ФК>	c_1	'И'	
<Компонент>	g_2	есть	<ЧастьРечи>	c_2	'И'	
<Компонент>	g_3	есть	<Окончание>	c_3	'И'	
<Компонент>	g_4	есть	<Падеж>	c_4	'И'	
<Компонент>	g_5	есть	<Число>	c_5	'И'	
<Имя>	n_1	имеет	<Значение>	["Словарь ДМИ"]	'И'	
<ЧастьРечи>	c_7	имеет	<Значение>	["сущ"]	'И'	
<Окончание>	c_3	эквивалентен	<Окончание>	o_1	'И'	
<Код ФК>	c_1	эквивалентен	<КодФК>	c_6	'И'	
<ЧастьРечи>	c_2	эквивалентен	<ЧастьРечи>	c_7		

ТО

<Лексема>	l_1	имеет	<Характеристика>	x_4	'И'
<Характеристика>	x_4	есть	<Падеж>	c_4	'И'
<Лексема>	l_1	имеет	<Характеристика>	x_5	'И'
<Характеристика>	x_5	есть	<Число>	c_5	

****

<*Продукционное правило* 17 на ограниченном подмножестве естественного языка>

ЕСЛИ

<Предложение>	p_1	содержит	<Лексема>	l_1	'И'
<Лексема>	l_1	содержит	<Окончание>	o_1	'И'
<Окончание>	o_1	принадлежит	<Словарь>	d_1	'И'
<Словарь>	d_1	содержит	<Вектор>	*Dmi*	'И'
<Вектор>	*Dmi*	содержит	<Компонент>	g_1	'И'
<Вектор>	*Dmi*	содержит	<Компонент>	g_2	'И'
<Вектор>	*Dmi*	содержит	<Компонент>	g_3	'И'
<Вектор>	*Dmi*	содержит	<Компонент>	g_4	'И'
<Вектор>	*Dmi*	содержит	<Компонент>	g_5	'И'
<Вектор>	*Dmi*	содержит	<Компонент>	g_6	'И'
<Словарь>	d_1	имеет	<Характеристика>	x_1	'И'
<Лексема>	l_1	имеет	<Характеристика>	x_2	'И'
<Лексема>	l_1	имеет	<Характеристика>	x_3	'И'
<Характеристика>	x_1	есть	<Имя>	n_1	'И'
<Характеристика>	x_2	есть	<КодФК>	c_7	'И'
<Характеристика>	x_3	есть	<ЧастьРечи>	c_8	'И'
<Компонент>	g_1	есть	<Код ФК>	c_1	'И'
<Компонент>	g_2	есть	<ЧастьРечи>	c_2	'И'
<Компонент>	g_3	есть	<Окончание>	c_3	'И'
<Компонент>	g_4	есть	<Падеж>	c_4	'И'
<Компонент>	g_5	есть	<Число>	c_5	'И'
<Компонент>	g_6	есть	<Род>	c_6	'И'
<Имя>	n_1	имеет	<Значение>	["Словарь ДМИ"]	'И'
<ЧастьРечи>	c_8	имеет	<Значение>	["прил"]	'И'
<ЧастьРечи>	c_3	эквивалентен	<Окончание>	o_1	'И'
<Код ФК>	c_1	эквивалентен	<КодФК>	c_7	'И'
<ЧастьРечи>	c_2	эквивалентен	<ЧастьРечи>	c_8	

ТО

<Лексема>	l_1	имеет	<Характеристика>	x_4	'И'
<Характеристика>	x_4	есть	<Падеж>	c_4	'И'
<Лексема>	l_1	имеет	<Характеристика>	x_5	'И'
<Характеристика>	x_5	есть	<Число>	c_5	'И'
<Лексема>	l_1	имеет	<Характеристика>	x_6	'И'
<Характеристика>	x_6	есть	<Число>	c_6	

****

<*Продукционное правило* 18 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит      <Лексема>        l1    'И'
<Лексема>        l1   содержит      <Окончание>      o1    'И'
<Окончание>      o1   принадлежит   <Словарь>        d1    'И'
<Словарь>        d1   содержит      <Вектор>         Dmi     'И'
<Вектор>         Dmi  содержит      <Компонент>      g1    'И'
<Вектор>         Dmi  содержит      <Компонент>      g2    'И'
<Вектор>         Dmi  содержит      <Компонент>      g3    'И'
<Вектор>         Dmi  содержит      <Компонент>      g4    'И'
<Вектор>         Dmi  содержит      <Компонент>      g5    'И'
<Словарь>        d1   имеет         <Характеристика> x1    'И'
<Лексема>        l1   имеет         <Характеристика> x2    'И'
<Лексема>        l1   имеет         <Характеристика> x3    'И'
<Характеристика> x1   есть          <Имя>            n1    'И'
<Характеристика> x2   есть          <КодФК>          c6    'И'
<Характеристика> x3   есть          <ЧастьРечи>      c7    'И'
<Компонент>      g1   есть          <Код ФК>         c1    'И'
<Компонент>      g2   есть          <ЧастьРечи>      c2    'И'
<Компонент>      g3   есть          <Окончание>      c3    'И'
<Компонент>      g4   есть          <Лицо>           c4    'И'
<Компонент>      g5   есть          <Число>          c5    'И'
<Имя>            n1   имеет         <Значение>       ["Словарь ДМИ"]'И'
<ЧастьРечи>      c6   имеет         <Значение>       ["глагол"]      'И'
<Окончание>      c3   эквивалентен  <Окончание>      o1    'И'
<Код ФК>         c1   эквивалентен  <КодФК>          c6    'И'
<ЧастьРечи>      c2   эквивалентен  <ЧастьРечи>      c7
ТО
<Лексема>        l1   имеет  <Характеристика>   x4   'И'
<Характеристика> x4   есть   <Лицо>            c4   'И'
<Лексема>        l1   имеет  <Характеристика>   x5   'И'
<Характеристика> x5   есть   <Число>            c5
```

4. Выделение словосочетаний

<*Продукционное правило* 1 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит      <Лексема>        l1    'И'
<Предложение>    p1   содержит      <Лексема>        l2    'И'
<Лексема>        l1   имеет         <Характеристика> x1    'И'
<Лексема>        l1   имеет         <Характеристика> x2    'И'
<Лексема>        l2   имеет         <Характеристика> x3    'И'
<Лексема>        l2   имеет         <Характеристика> x4    'И'
<Характеристика> x1   есть          <ЧастьРечи>      c1    'И'
<Характеристика> x2   есть          <Падеж>          c2    'И'
<Характеристика> x3   есть          <ЧастьРечи>      c3    'И'
<Характеристика> x4   есть          <Падеж>          c4    'И'
<ЧастьРечи>      c1   имеет         <Значение>       ["Прил"]  'И'
<ЧастьРечи>      c3   имеет         <Значение>       ["Сущ"]   'И'
<Падеж>          c2   эквивалентен  <Падеж>          c4    'И'
<Лексема>        l1   имеет         <Индекс>         i     'И'
<Лексема>        l2   имеет         <Индекс>         (i+1)
ТО
<Лексема> l1 и <Лексема> l2  составляют <Категория>  ["ИСС"]
```

****

<*Продукционное правило* 2 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит      <Лексема>        l1    'И'
<Предложение>    p1   содержит      <Лексема>        l2    'И'
<Предложение>    p1   содержит      <Лексема>        l3    'И'
<Лексема>        l1   имеет         <Характеристика> x1    'И'
```

<Лексема> l_1 имеет <Характеристика> x_2 'И'
<Лексема> l_2 имеет <Характеристика> x_3 'И'
<Лексема> l_2 имеет <Характеристика> x_4 'И'
<Лексема> l_3 имеет <Характеристика> x_5 'И'
<Лексема> l_3 имеет <Характеристика> x_6 'И'
<Характеристика> x_1 есть <ЧастьРечи> c_1 'И'
<Характеристика> x_2 есть <Падеж> c_2 'И'
<Характеристика> x_3 есть <ЧастьРечи> c_3 'И'
<Характеристика> x_4 есть <Падеж> c_4 'И'
<Характеристика> x_5 есть <ЧастьРечи> c_5 'И'
<Характеристика> x_6 есть <Падеж> c_6 'И'
<ЧастьРечи> c_1 имеет <Значение> ["Прил"] 'И'
<ЧастьРечи> c_3 имеет <Значение> ["Прил"] 'И'
<ЧастьРечи> c_5 имеет <Значение> ["Сущ"] 'И'
<Падеж> c_2 эквивалентен <Падеж> c_4 'И'
<Падеж> c_4 эквивалентен <Падеж> c_6 'И'
<Падеж> c_6 эквивалентен <Падеж> c_2 'И'
<Лексема> l_1 имеет <Индекс> i 'И'
<Лексема> l_2 имеет <Индекс> (i+1) 'И'
<Лексема> l_3 имеет <Индекс> (i+2)
ТО
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_3 составляют <Категория> ["ИСС"]

****

<*Продукционное правило* 3 на ограниченном подмножестве естественного языка>

ЕСЛИ
<Предложение> p_1 содержит <Лексема> l_1 'И'
<Предложение> p_1 содержит <Лексема> l_2 'И'
<Предложение> p_1 содержит <Лексема> l_3 'И'
<Предложение> p_1 содержит <Лексема> l_4 'И'
<Лексема> l_1 имеет <Характеристика> x_1 'И'
<Лексема> l_1 имеет <Характеристика> x_2 'И'
<Лексема> l_2 имеет <Характеристика> x_3 'И'
<Лексема> l_2 имеет <Характеристика> x_4 'И'
<Лексема> l_3 имеет <Характеристика> x_5 'И'
<Лексема> l_3 имеет <Характеристика> x_6 'И'
<Лексема> l_4 имеет <Характеристика> x_7 'И'
<Лексема> l_4 имеет <Характеристика> x_8 'И'
<Характеристика> x_1 есть <ЧастьРечи> c_1 'И'
<Характеристика> x_2 есть <Падеж> c_2 'И'
<Характеристика> x_3 есть <ЧастьРечи> c_3 'И'
<Характеристика> x_4 есть <Падеж> c_4 'И'
<Характеристика> x_5 есть <ЧастьРечи> c_5 'И'
<Характеристика> x_6 есть <Падеж> c_6 'И'
<Характеристика> x_7 есть <ЧастьРечи> c_7 'И'
<Характеристика> x_8 есть <Падеж> c_8 'И'
<ЧастьРечи> c_1 имеет <Значение> ["Прил"] 'И'
<ЧастьРечи> c_3 имеет <Значение> ["Прил"] 'И'
<ЧастьРечи> c_5 имеет <Значение> ["Прил"] 'И'
<ЧастьРечи> c_7 имеет <Значение> ["Сущ"] 'И'
<Падеж> c_2 эквивалентен <Падеж> c_4 'И'
<Падеж> c_4 эквивалентен <Падеж> c_6 'И'
<Падеж> c_6 эквивалентен <Падеж> c_8 'И'
<Падеж> c_8 эквивалентен <Падеж> c_2 'И'
<Лексема> l_1 имеет <Индекс> i 'И'
<Лексема> l_2 имеет <Индекс> (i+1) 'И'
<Лексема> l_3 имеет <Индекс> (i+2) 'И'
<Лексема> l_4 имеет <Индекс> (i+3)
ТО
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_3 и <Лексема> l_4 составляют <Категория> ["ИСС"]

****

<*Продукционное правило* 4 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит  <Лексема>        l1      'И'
<Предложение>    p1   содержит  <Лексема>        l2      'И'
<Лексема>        l1   имеет     <Характеристика> x1      'И'
<Лексема>        l1   имеет     <Характеристика> x2      'И'
<Лексема>        l2   имеет     <Характеристика> x3      'И'
<Лексема>        l2   имеет     <Характеристика> x4      'И'
<Характеристика> x1   есть      <ЧастьРечи>     c1      'И'
<Характеристика> x2   есть      <Падеж>         c2      'И'
<Характеристика> x3   есть      <ЧастьРечи>     c3      'И'
<Характеристика> x4   есть      <Падеж>         c4      'И'
<ЧастьРечи>      c1   имеет     <Значение>      ["Сущ"] 'И'
<ЧастьРечи>      c3   имеет     <Значение>      ["Сущ"] 'И'
<Падеж>          c2   имеет     <Значение>      ["Им"]  'И'
<Падеж>          c4   имеет     <Значение>      ["Род"] 'И'
<Лексема>        l1   имеет     <Индекс>        i       'И'
<Лексема>        l2   имеет     <Индекс>        (i+1)
ТО
<Лексема> l1   и <Лексема> l2 составляют <Категория> ["ИСС"]
**********************************************************************************
```

<*Продукционное правило* 5 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит  <Лексема>        l1    'И'
<Предложение>    p1   содержит  <Лексема>        l2    'И'
<Предложение>    p1   содержит  <Лексема>        l3    'И'
<Лексема>        l1   имеет     <Характеристика> x1    'И'
<Лексема>        l1   имеет     <Характеристика> x2    'И'
<Лексема>        l2   имеет     <Характеристика> x3    'И'
<Лексема>        l2   имеет     <Характеристика> x4    'И'
<Лексема>        l3   имеет     <Характеристика> x5    'И'
<Лексема>        l3   имеет     <Характеристика> x6    'И'
<Характеристика> x1   есть      <ЧастьРечи>     c1    'И'
<Характеристика> x2   есть      <Падеж>         c2    'И'
<Характеристика> x3   есть      <ЧастьРечи>     c3    'И'
<Характеристика> x4   есть      <Падеж>         c4    'И'
<Характеристика> x5   есть      <ЧастьРечи>     c5    'И'
<Характеристика> x6   есть      <Падеж>         c6    'И'
<ЧастьРечи>      c1   имеет     <Значение>      ["Сущ"]   'И'
<ЧастьРечи>      c3   имеет     <Значение>      ["Прил"]  'И'
<ЧастьРечи>      c5   имеет     <Значение>      ["Сущ"]   'И'
<Падеж>          c2   имеет     <Значение>      ["Им"]    'И'
<Падеж>          c4   имеет     <Значение>      ["Род"]   'И'
<Падеж>          c6   имеет     <Значение>      ["Род"]   'И'
<Лексема>        l1   имеет     <Индекс>        i         'И'
<Лексема>        l2   имеет     <Индекс>        (i+1)     'И'
<Лексема>        l3   имеет     <Индекс>        (i+2)
ТО
<Лексема> l1 и <Лексема> l2 и <Лексема> l3 составляют <Категория> ["КСС"] 'И'
<Лексема> l2 и <Лексема> l3  составляют <Категория> ["ИСС"]
**********************************************************************************
```

<*Продукционное правило* 6 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит  <Лексема>        l1    'И'
<Предложение>    p1   содержит  <Лексема>        l2    'И'
<Предложение>    p1   содержит  <Лексема>        l3    'И'
<Предложение>    p1   содержит  <Лексема>        l4    'И'
<Лексема>        l1   имеет     <Характеристика> x1    'И'
<Лексема>        l1   имеет     <Характеристика> x2    'И'
<Лексема>        l2   имеет     <Характеристика> x3    'И'
```

```
<Лексема>         l2   имеет  <Характеристика> x4   'И'
<Лексема>         l3   имеет  <Характеристика> x5   'И'
<Лексема>         l3   имеет  <Характеристика> x6   'И'
<Лексема>         l4   имеет  <Характеристика> x7   'И'
<Лексема>         l4   имеет  <Характеристика> x8   'И'
<Характеристика> x1   есть   <ЧастьРечи>      c1   'И'
<Характеристика> x2   есть   <Падеж>          c2   'И'
<Характеристика> x3   есть   <ЧастьРечи>      c3   'И'
<Характеристика> x4   есть   <Падеж>          c4   'И'
<Характеристика> x5   есть   <ЧастьРечи>      c5   'И'
<Характеристика> x6   есть   <Падеж>          c6   'И'
<Характеристика> x7   есть   <ЧастьРечи>      c7   'И'
<Характеристика> x8   есть   <Падеж>          c8   'И'
<ЧастьРечи>       c1   имеет  <Значение>       ["Сущ"]  'И'
<ЧастьРечи>       c3   имеет  <Значение>       ["Прил"] 'И'
<ЧастьРечи>       c5   имеет  <Значение>       ["Прил"] 'И'
<ЧастьРечи>       c7   имеет  <Значение>       ["Сущ"]  'И'
<Падеж>           c2   имеет  <Значение>       ["Им"]   'И'
<Падеж>           c4   имеет  <Значение>       ["Род"]  'И'
<Падеж>           c6   имеет  <Значение>       ["Род"]  'И'
<Падеж>           c8   имеет  <Значение>       ["Род"]  'И'
<Лексема>         l1   имеет  <Индекс>         i        'И'
<Лексема>         l2   имеет  <Индекс>         (i+1)    'И'
<Лексема>         l3   имеет  <Индекс>         (i+2)    'И'
<Лексема>         l4   имеет  <Индекс>         (i+3)
ТО
<Лексема> l1 и <Лексема> l2 и <Лексема> l3  и <Лексема> l4 составляют
<Категория> ["КСС"]'И'
<Лексема> l2 и <Лексема> l3 и <Лексема> l4    составляют <Категория> ["ИСС"]
*******************************************************************************
```

<*Продукционное правило* 7 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит <Лексема>         l1   'И'
<Предложение>    p1   содержит <Лексема>         l2   'И'
<Предложение>    p1   содержит <Лексема>         l3   'И'
<Предложение>    p1   содержит <Лексема>         l4   'И'
<Предложение>    p1   содержит <Лексема>         l5   'И'
<Лексема>        l1   имеет    <Характеристика> x1   'И'
<Лексема>        l1   имеет    <Характеристика> x2   'И'
<Лексема>        l2   имеет    <Характеристика> x3   'И'
<Лексема>        l2   имеет    <Характеристика> x4   'И'
<Лексема>        l3   имеет    <Характеристика> x5   'И'
<Лексема>        l3   имеет    <Характеристика> x6   'И'
<Лексема>        l4   имеет    <Характеристика> x7   'И'
<Лексема>        l4   имеет    <Характеристика> x8   'И'
<Лексема>        l5   имеет    <Характеристика> x9   'И'
<Лексема>        l5   имеет    <Характеристика> x10  'И'
<Характеристика> x1   есть     <ЧастьРечи>      c1   'И'
<Характеристика> x2   есть     <Падеж>          c2   'И'
<Характеристика> x3   есть     <ЧастьРечи>      c3   'И'
<Характеристика> x4   есть     <Падеж>          c4   'И'
<Характеристика> x5   есть     <ЧастьРечи>      c5   'И'
<Характеристика> x6   есть     <Падеж>          c6   'И'
<Характеристика> x7   есть     <ЧастьРечи>      c7   'И'
<Характеристика> x8   есть     <Падеж>          c8   'И'
<Характеристика> x9   есть     <ЧастьРечи>      c9   'И'
<Характеристика> x10  есть     <Падеж>          c10  'И'
<ЧастьРечи>      c1   имеет    <Значение>  ["Сущ"]   'И'
<ЧастьРечи>      c3   имеет    <Значение>  ["Прил"]  'И'
<ЧастьРечи>      c5   имеет    <Значение>  ["Прил"]  'И'
<ЧастьРечи>      c7   имеет    <Значение>  ["Прил"]  'И'
<ЧастьРечи>      c9   имеет    <Значение>  ["Сущ"]   'И'
```

<Падеж> c_2 имеет <Значение> [“Им”] ‘И’
<Падеж> c_4 имеет <Значение> [“Род”] ‘И’
<Падеж> c_6 имеет <Значение> [“Род”] ‘И’
<Падеж> c_8 имеет <Значение> [“Род”] ‘И’
<Падеж> c_{10} имеет <Значение> [“Род”] ‘И’
<Лексема> l_1 имеет <Индекс> i ‘И’
<Лексема> l_2 имеет <Индекс> (i+1) ‘И’
<Лексема> l_3 имеет <Индекс> (i+2) ‘И’
<Лексема> l_4 имеет <Индекс> (i+3) ‘И’
<Лексема> l_5 имеет <Индекс> (i+4)
ТО
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_3 и <Лексема> l_4 и <Лексема> l_5 составляют <Категория> [“КСС”] ‘И’
<Лексема> l_2 и <Лексема> l_3 и <Лексема> l_4 и <Лексема> l_5 составляют <Категория> [“ИСС”]

****

<*Продукционное правило* 8 на ограниченном подмножестве естественного языка>

ЕСЛИ
<Предложение> p_1 содержит <Лексема> l_1 ‘И’
<Предложение> p_1 содержит <Лексема> l_2 ‘И’
<Предложение> p_1 содержит <Лексема> l_3 ‘И’
<Лексема> l_1 имеет <Характеристика> x_1 ‘И’
<Лексема> l_1 имеет <Характеристика> x_2 ‘И’
<Лексема> l_2 имеет <Характеристика> x_3 ‘И’
<Лексема> l_2 имеет <Характеристика> x_4 ‘И’
<Лексема> l_3 имеет <Характеристика> x_5 ‘И’
<Лексема> l_3 имеет <Характеристика> x_6 ‘И’
<Характеристика> x_1 есть <ЧастьРечи> c_1 ‘И’
<Характеристика> x_2 есть <Падеж> c_2 ‘И’
<Характеристика> x_3 есть <ЧастьРечи> c_3 ‘И’
<Характеристика> x_4 есть <Падеж> c_4 ‘И’
<Характеристика> x_5 есть <ЧастьРечи> c_5 ‘И’
<Характеристика> x_6 есть <Падеж> c_6 ‘И’
<ЧастьРечи> c_1 имеет <Значение> [“Прил”] ‘И’
<ЧастьРечи> c_3 имеет <Значение> [“Сущ”] ‘И’
<ЧастьРечи> c_5 имеет <Значение> [“Сущ”] ‘И’
<Падеж> c_2 имеет <Значение> [“Им”] ‘И’
<Падеж> c_4 имеет <Значение> [“Им”] ‘И’
<Падеж> c_6 имеет <Значение> [“Род”] ‘И’
<Лексема> l_1 имеет <Индекс> i ‘И’
<Лексема> l_2 имеет <Индекс> (i+1) ‘И’
<Лексема> l_3 имеет <Индекс> (i+2)
ТО
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_3 составляют <Категория> [“КСС”] ‘И’
<Лексема> l_1 и <Лексема> l_2 составляют <Категория> [“ИСС”]

****

<*Продукционное правило* 9 на ограниченном подмножестве естественного языка>

ЕСЛИ
<Предложение> p_1 содержит <Лексема> l_1 ‘И’
<Предложение> p_1 содержит <Лексема> l_2 ‘И’
<Предложение> p_1 содержит <Лексема> l_3 ‘И’
<Предложение> p_1 содержит <Лексема> l_4 ‘И’
<Лексема> l_1 имеет <Характеристика> x_1 ‘И’
<Лексема> l_1 имеет <Характеристика> x_2 ‘И’
<Лексема> l_2 имеет <Характеристика> x_3 ‘И’
<Лексема> l_2 имеет <Характеристика> x_4 ‘И’
<Лексема> l_3 имеет <Характеристика> x_5 ‘И’
<Лексема> l_3 имеет <Характеристика> x_6 ‘И’
<Лексема> l_4 имеет <Характеристика> x_7 ‘И’
<Лексема> l_4 имеет <Характеристика> x_8 ‘И’

```
<Характеристика> x1   есть   <ЧастьРечи>        c1    'И'
<Характеристика> x2   есть   <Падеж>            c2    'И'
<Характеристика> x3   есть   <ЧастьРечи>        c3    'И'
<Характеристика> x4   есть   <Падеж>            c4    'И'
<Характеристика> x5   есть   <ЧастьРечи>        c5    'И'
<Характеристика> x6   есть   <Падеж>            c6    'И'
<Характеристика> x7   есть   <ЧастьРечи>        c7    'И'
<Характеристика> x8   есть   <Падеж>            c8    'И'
<ЧастьРечи>      c1   имеет  <Значение>         ["Прил"]  'И'
<ЧастьРечи>      c3   имеет  <Значение>         ["Сущ"]   'И'
<ЧастьРечи>      c5   имеет  <Значение>         ["Сущ"]   'И'
<ЧастьРечи>      c7   имеет  <Значение>         ["Сущ"]   'И'
<Падеж>          c2   имеет  <Значение>         ["Им"]    'И'
<Падеж>          c4   имеет  <Значение>         ["Им"]    'И'
<Падеж>          c6   имеет  <Значение>         ["Род"]   'И'
<Падеж>          c8   имеет  <Значение>         ["Род"]   'И'
<Лексема>        l1   имеет  <Индекс>           i         'И'
<Лексема>        l2   имеет  <Индекс>           (i+1)     'И'
<Лексема>        l3   имеет  <Индекс>           (i+2)     'И'
<Лексема>        l4   имеет  <Индекс>           (i+3)
ТО
<Лексема>      l1 и <Лексема> l2 и <Лексема> l3и <Лексема> l4 составляют
<Категория> ["КСС"]'И'
<Лексема>      l1 и <Лексема> l2   составляют <Категория> ["ИСС"]'И'
<Лексема>      l3 и <Лексема> l4   составляют <Категория> ["ИСС"]
```

****

<*Продукционное правило* 10 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит  <Лексема>         l1    'И'
<Предложение>    p1   содержит  <Лексема>         l2    'И'
<Предложение>    p1   содержит  <Лексема>         l3    'И'
<Предложение>    p1   содержит  <Лексема>         l4    'И'
<Предложение>    p1   содержит  <Лексема>         l5    'И'
<Лексема>        l1   имеет     <Характеристика>  x1    'И'
<Лексема>        l1   имеет     <Характеристика>  x2    'И'
<Лексема>        l2   имеет     <Характеристика>  x3    'И'
<Лексема>        l2   имеет     <Характеристика>  x4    'И'
<Лексема>        l3   имеет     <Характеристика>  x5    'И'
<Лексема>        l3   имеет     <Характеристика>  x6    'И'
<Лексема>        l4   имеет     <Характеристика>  x7    'И'
<Лексема>        l4   имеет     <Характеристика>  x8    'И'
<Лексема>        l5   имеет     <Характеристика>  x9    'И'
<Лексема>        l5   имеет     <Характеристика>  x10   'И'
<Характеристика> x1   есть      <ЧастьРечи>      c1    'И'
<Характеристика> x2   есть      <Падеж>          c2    'И'
<Характеристика> x3   есть      <ЧастьРечи>      c3    'И'
<Характеристика> x4   есть      <Падеж>          c4    'И'
<Характеристика> x5   есть      <ЧастьРечи>      c5    'И'
<Характеристика> x6   есть      <Падеж>          c6    'И'
<Характеристика> x7   есть      <ЧастьРечи>      c7    'И'
<Характеристика> x8   есть      <Падеж>          c8    'И'
<Характеристика> x9   есть      <ЧастьРечи>      c9    'И'
<Характеристика> x10  есть      <Падеж>          c10   'И'
<ЧастьРечи>      c1   имеет     <Значение>       ["Прил"]  'И'
<ЧастьРечи>      c3   имеет     <Значение>       ["Прил"]  'И'
<ЧастьРечи>      c5   имеет     <Значение>       ["Сущ"]   'И'
<ЧастьРечи>      c7   имеет     <Значение>       ["Прил"]  'И'
<ЧастьРечи>      c9   имеет     <Значение>       ["Сущ"]   'И'
<Падеж>          c2   имеет     <Значение>       ["Им"]    'И'
<Падеж>          c4   имеет     <Значение>       ["Им"]    'И'
<Падеж>          c6   имеет     <Значение>       ["Им"]    'И'
<Падеж>          c8   имеет     <Значение>       ["Род"]   'И'
```

<Падеж> c_{10} имеет <Значение> ["Род"] 'И'
<Лексема> l_1 имеет <Индекс> i 'И'
<Лексема> l_2 имеет <Индекс> (i+1) 'И'
<Лексема> l_3 имеет <Индекс> (i+2) 'И'
<Лексема> l_4 имеет <Индекс> (i+3) 'И'
<Лексема> l_5 имеет <Индекс> (i+4)
ТО
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_3 и <Лексема> l_4 и <Лексема> l_5 составляют <Категория> ["КСС"] 'И'
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_3 составляют <Категория> ["ИСС"] 'И'
<Лексема> l_4 и <Лексема> l_5 составляют <Категория> ["ИСС"]

****

<*Продукционное правило* 11 на ограниченном подмножестве естественного языка>

ЕСЛИ
<Предложение> p_1 содержит <Лексема> l_1 'И'
<Предложение> p_1 содержит <Лексема> l_2 'И'
<Предложение> p_1 содержит <Лексема> l_3 'И'
<Лексема> l_1 имеет <Характеристика> x_1 'И'
<Лексема> l_1 имеет <Характеристика> x_2 'И'
<Лексема> l_2 имеет <Характеристика> x_3 'И'
<Лексема> l_2 имеет <Характеристика> x_4 'И'
<Лексема> l_3 имеет <Характеристика> x_5 'И'
<Лексема> l_3 имеет <Характеристика> x_6 'И'
<Характеристика> x_1 есть <ЧастьРечи> c_1 'И'
<Характеристика> x_2 есть <Падеж> c_2 'И'
<Характеристика> x_3 есть <ЧастьРечи> c_3 'И'
<Характеристика> x_4 есть <Падеж> c_4 'И'
<Характеристика> x_5 есть <ЧастьРечи> c_5 'И'
<Характеристика> x_6 есть <Падеж> c_6 'И'
<ЧастьРечи> c_1 имеет <Значение> ["Сущ"] 'И'
<ЧастьРечи> c_3 имеет <Значение> ["Сущ"] 'И'
<ЧастьРечи> c_5 имеет <Значение> ["Сущ"] 'И'
<Падеж> c_2 имеет <Значение> ["Им"] 'И'
<Падеж> c_4 имеет <Значение> ["Род"] 'И'
<Падеж> c_6 имеет <Значение> ["Род"] 'И'
<Лексема> l_1 имеет <Индекс> i 'И'
<Лексема> l_2 имеет <Индекс> (i+1) 'И'
<Лексема> l_3 имеет <Индекс> (i+2)
ТО
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_3 составляют <Категория> ["КСС"] 'И'
<Лексема> l_2 и <Лексема> l_3 составляют <Категория> ["ИСС"]

****

<*Продукционное правило* 12 на ограниченном подмножестве естественного языка>

ЕСЛИ
<Предложение> p_1 содержит <Лексема> l_1 'И'
<Предложение> p_1 содержит <Лексема> l_2 'И'
<Предложение> p_1 содержит <Лексема> l_3 'И'
<Предложение> p_1 содержит <Лексема> l_4 'И'
<Лексема> l_1 имеет <Характеристика> x_1 'И'
<Лексема> l_1 имеет <Характеристика> x_2 'И'
<Лексема> l_2 имеет <Характеристика> x_3 'И'
<Лексема> l_2 имеет <Характеристика> x_4 'И'
<Лексема> l_3 имеет <Характеристика> x_5 'И'
<Лексема> l_3 имеет <Характеристика> x_6 'И'
<Лексема> l_4 имеет <Характеристика> x_7 'И'
<Лексема> l_4 имеет <Характеристика> x_8 'И'
<Характеристика> x_1 есть <ЧастьРечи> c_1 'И'
<Характеристика> x_2 есть <Падеж> c_2 'И'
<Характеристика> x_3 есть <ЧастьРечи> c_3 'И'
<Характеристика> x_4 есть <Падеж> c_4 'И'

```
<Характеристика> x5   есть     <ЧастьРечи>      c5 'И'
<Характеристика> x6   есть     <Падеж>          c6 'И'
<Характеристика> x7   есть     <ЧастьРечи>      c7 'И'
<Характеристика> x8   есть     <Падеж>          c8 'И'
<ЧастьРечи>      c1   имеет    <Значение>       ["Сущ"] 'И'
<ЧастьРечи>      c3   имеет    <Значение>       ["Сущ"] 'И'
<ЧастьРечи>      c5   имеет    <Значение>       ["Сущ"] 'И'
<ЧастьРечи>      c7   имеет    <Значение>       ["Сущ"] 'И'
<Падеж>          c2   имеет    <Значение>       ["Им"]  'И'
<Падеж>          c4   имеет    <Значение>       ["Род"] 'И'
<Падеж>          c6   имеет    <Значение>       ["Тв"]  'И'
<Падеж>          c8   имеет    <Значение>       ["Род"] 'И'
<Лексема>        l1   имеет    <Индекс>         i       'И'
<Лексема>        l2   имеет    <Индекс>         (i+1)   'И'
<Лексема>        l3   имеет    <Индекс>         (i+2)   'И'
<Лексема>        l4   имеет    <Индекс>         (i+3)
ТО
<Лексема>      l1 и <Лексема> l2 и <Лексема> l3и <Лексема> l4 составляют
<Категория> ["КСС"]'И'
<Лексема>      l1 и <Лексема> l2   составляют <Категория> ["ИСС"]'И'
<Лексема>      l3 и <Лексема> l4   составляют <Категория> ["ИСС"]
```

****

<*Продукционное правило* 13 на ограниченном подмножестве естественного языка>

```
ЕСЛИ
<Предложение>    p1   содержит  <Лексема>        l1    'И'
<Предложение>    p1   содержит  <Лексема>        l2    'И'
<Предложение>    p1   содержит  <Лексема>        l3    'И'
<Предложение>    p1   содержит  <Лексема>        l4    'И'
<Предложение>    p1   содержит  <Лексема>        l5    'И'
<Лексема>        l1   имеет     <Характеристика> x1    'И'
<Лексема>        l1   имеет     <Характеристика> x2    'И'
<Лексема>        l2   имеет     <Характеристика> x3    'И'
<Лексема>        l2   имеет     <Характеристика> x4    'И'
<Лексема>        l3   имеет     <Характеристика> x5    'И'
<Лексема>        l4   имеет     <Характеристика> x6    'И'
<Лексема>        l4   имеет     <Характеристика> x7    'И'
<Лексема>        l5   имеет     <Характеристика> x8    'И'
<Лексема>        l5   имеет     <Характеристика> x9    'И'
<Характеристика> x1   есть      <ЧастьРечи>      c1    'И'
<Характеристика> x2   есть      <Падеж>          c2    'И'
<Характеристика> x3   есть      <ЧастьРечи>      c3    'И'
<Характеристика> x4   есть      <Падеж>          c4    'И'
<Характеристика> x5   есть      <ЧастьРечи>      c5    'И'
<Характеристика> x6   есть      <ЧастьРечи>      c6    'И'
<Характеристика> x7   есть      <Падеж>          c7    'И'
<Характеристика> x8   есть      <ЧастьРечи>      c8    'И'
<Характеристика> x9   есть      <Падеж>          c9    'И'
<ЧастьРечи>      c1   имеет     <Значение>       ["Сущ"] 'И'
<ЧастьРечи>      c3   имеет     <Значение>       ["Сущ"] 'И'
<ЧастьРечи>      c5   имеет     <Значение>       ["Союз"]'И'
<ЧастьРечи>      c6   имеет     <Значение>       ["Сущ"] 'И'
<ЧастьРечи>      c8   имеет     <Значение>       ["Сущ"] 'И'
<Лексема>        l3   имеет     <Значение>       ['и']   'И'
<Падеж>          c2   имеет     <Значение>       ["Им"]  'И'
<Падеж>          c4   имеет     <Значение>       ["Род"] 'И'
<Падеж>          c7   имеет     <Значение>       ["Род"] 'И'
<Падеж>          c9   имеет     <Значение>       ["Род"] 'И'
<Лексема>        l1   имеет     <Индекс>         i       'И'
<Лексема>        l2   имеет     <Индекс>         (i+1)   'И'
<Лексема>        l3   имеет     <Индекс>         (i+2)   'И'
<Лексема>        l4   имеет     <Индекс>         (i+3)   'И'
<Лексема>        l5   имеет     <Индекс>         (i+4)   'И'
```

ТО
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_3 и <Лексема> l_4 и <Лексема> l_5
составляют <Категория> ["КСС"]'И'
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_5 составляют <Категория> ["ИСС"]'И'
<Лексема> l_1 и <Лексема> l_4 и <Лексема> l_5 составляют <Категория> ["ИСС"]

****

<*Продукционное правило* 14 на ограниченном подмножестве естественного языка>

ЕСЛИ

<Предложение>	p_1	содержит	<Лексема>	l_1	'И'
<Предложение>	p_1	содержит	<Лексема>	l_2	'И'
<Предложение>	p_1	содержит	<Лексема>	l_3	'И'
<Предложение>	p_1	содержит	<Лексема>	l_4	'И'
<Предложение>	p_1	содержит	<Лексема>	l_5	'И'
<Лексема>	l_1	имеет	<Характеристика>	x_1	'И'
<Лексема>	l_1	имеет	<Характеристика>	x_2	'И'
<Лексема>	l_2	имеет	<Характеристика>	x_3	'И'
<Лексема>	l_2	имеет	<Характеристика>	x_4	'И'
<Лексема>	l_3	имеет	<Характеристика>	x_5	'И'
<Лексема>	l_3	имеет	<Характеристика>	x_6	'И'
<Лексема>	l_4	имеет	<Характеристика>	x_7	'И'
<Лексема>	l_5	имеет	<Характеристика>	x_8	'И'
<Лексема>	l_5	имеет	<Характеристика>	x_9	'И'
<Характеристика>	x_1	есть	<ЧастьРечи>	c_1	'И'
<Характеристика>	x_2	есть	<Падеж>	c_2	'И'
<Характеристика>	x_3	есть	<ЧастьРечи>	c_3	'И'
<Характеристика>	x_4	есть	<Падеж>	c_4	'И'
<Характеристика>	x_5	есть	<ЧастьРечи>	c_5	'И'
<Характеристика>	x_6	есть	<Падеж>	c_6	'И'
<Характеристика>	x_7	есть	<ЧастьРечи>	c_7	'И'
<Характеристика>	x_8	есть	<ЧастьРечи>	c_8	'И'
<Характеристика>	x_9	есть	<Падеж>	c_9	'И'
<ЧастьРечи>	c_1	имеет	<Значение>	["Прил"]	'И'
<ЧастьРечи>	c_3	имеет	<Значение>	["Сущ"]	'И'
<ЧастьРечи>	c_5	имеет	<Значение>	["Сущ"]	'И'
<ЧастьРечи>	c_6	имеет	<Значение>	["Союз"]	'И'
<ЧастьРечи>	c_8	имеет	<Значение>	["Сущ"]	'И'
<Лексема>	l_4	имеет	<Значение>	['и']	'И'
<Падеж>	c_2	имеет	<Значение>	["Им"]	'И'
<Падеж>	c_4	имеет	<Значение>	["Им"]	'И'
<Падеж>	c_7	имеет	<Значение>	["Род"]	'И'
<Падеж>	c_9	имеет	<Значение>	["Род"]	'И'
<Лексема>	l_1	имеет	<Индекс>	i	'И'
<Лексема>	l_2	имеет	<Индекс>	(i+1)	'И'
<Лексема>	l_3	имеет	<Индекс>	(i+2)	'И'
<Лексема>	l_4	имеет	<Индекс>	(i+3)	'И'
<Лексема>	l_5	имеет	<Индекс>	(i+4)	

ТО
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_3 и <Лексема> l_4 и <Лексема> l_5
составляют <Категория> ["КСС"]'И'
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_5 составляют <Категория> ["ИСС"]'И'
<Лексема> l_1 и <Лексема> l_2 и <Лексема> l_5 составляют <Категория> ["ИСС"]

ПРИЛОЖЕНИЕ В

Примеры словарных статей терминосистемы и номенклатуры
в формате XML

1. Примеры словарных статей терминосистемы

```
<FRAME NAME = "FrameTS_10.xml" >
  <NAMETERM VALUE = "Заработная плата" >
  <SLOT NAME = "NameCO" VALUE = "Понятие" >
  <SLOT NAME = "KindEntity" VALUE = "Материальный" >
  <SLOT NAME = "definitions" >
    <SLOT NAME = "definition" VALUE = "денежное вознаграждение, выплачиваемое работодателем за выполненный работником труд" >
    <SLOT NAME = "attachproc" VALUE = "FunDef" >
  <SLOT NAME = "propeties" >
    <SLOT NAME = "property" >
      <SLOT NAME = "nameprop" VALUE = "размер минимальной заработной платы" >
      <SLOT NAME = "link" FILE = "Frame_11.xml" >
    <SLOT NAME = "property" >
      <SLOT NAME = "nameprop" VALUE = "единая тарифная сетка" >
      <SLOT NAME = "link" FILE = "Frame_#.xml" >
    <SLOT NAME = "attachproc" VALUE = "FunProp" >
  <SLOT NAME = "actions" >
    <SLOT NAME = "action" >
      <SLOT NAME = "nameact" VALUE = "регулируется" >
      <SLOT NAME = "link" FILE = "Frame_#.xml" >
    <SLOT NAME = "listattachproc" >
      <SLOT NAME = "attachproc" VALUE = "FunAct" >
      <SLOT NAME = "attachproc" VALUE = "ActivProc" >
  <SLOT NAME = "kvantrel" >
    <SLOT NAME = "synonyms" >
      <SLOT NAME = "synonym" >
        <SLOT NAME = "namesyn" VALUE = "жалованье" >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
      <SLOT NAME = "synonym" >
        <SLOT NAME = "namesyn" VALUE = "денежное содержание" >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
      <SLOT NAME = "synonym" >
        <SLOT NAME = "namesyn" VALUE = "должностной оклад" >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
      <SLOT NAME = "synonym" >
        <SLOT NAME = "namesyn" VALUE = "заработок" >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
      <SLOT NAME = "attachproc" VALUE = "FunSyn" >
    <SLOT NAME = "correlates" >
  <SLOT NAME = "states" >
  <SLOT NAME = "kvalitrel" >
    <SLOT NAME = "ClassKind" >
      <SLOT NAME = "Class" >
        <SLOT NAME = "nameclass" VALUE = "денежное вознаграждение" >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunClass" >
      <SLOT NAME = "Kind" >
        <SLOT NAME = "namekind" VALUE = "сдельная заработная плата" >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
        <SLOT NAME = "namekind" VALUE = "повременная заработная плата" >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
        <SLOT NAME = "namekind" VALUE = "аккордная заработная плата" >
        <SLOT NAME = "link" FILE = "Frame_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunKind" >
    <SLOT NAME = "WholePart" >
    <SLOT NAME = "PresMethods" >
    <SLOT NAME = "Styles" >
  <SLOT NAME = "metasign" >
  <SLOT NAME = "listattachproc" >
```

****

```
<FRAME NAME = "FrameTS_11.xml" >
    <NAMETERM VALUE = "размер минимальной заработной платы" >
    <SLOT NAME = "NameCO" VALUE = "Величина" >
    <SLOT NAME = "TypeQuantity" VALUE = "цена" >
  + <SLOT NAME = "definitions" >
  + <SLOT NAME = "MeasurementScale" >
  + <SLOT NAME = "InterOfValues" >
    <SLOT NAME = "UnitMeasure" VALUE = "..." >
  + <SLOT NAME = "MetaSign" >
  + <SLOT NAME = "ListAttachProc" >
```

****

```
- <FRAME NAME = "FrameTS_12.xml" >
    <NAMETERM VALUE = "единая тарифная сетка" >
    <SLOT NAME = "NameCO" VALUE = "Понятие" >
    <SLOT NAME = "KindEntity" VALUE = "Материальный" >
  - <SLOT NAME = "definitions" >
      <SLOT NAME = "definition" VALUE = "система разрядов, служащая для определения правильных соотношений между оплатой труда и квалиф
      <SLOT NAME = "attachproc" VALUE = "FunDef" >
  + <SLOT NAME = "propeties" >
  + <SLOT NAME = "actions" >
  + <SLOT NAME = "kvantrel" >
  + <SLOT NAME = "states" >
  - <SLOT NAME = "kvalitrel" >
    + <SLOT NAME = "ClassKind" >
    - <SLOT NAME = "WholePart" >
      - <SLOT NAME = "Whole" >
          <SLOT NAME = "namewhole" VALUE = "тарифная система" >
          <SLOT NAME = "link" FILE = "Frame_#.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunWhole" >
      - <SLOT NAME = "Part" >
          <SLOT NAME = "namepart" VALUE = "разряд" >
          <SLOT NAME = "link" FILE = "Frame_#.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunPart" >
    + <SLOT NAME = "PresMethods" >
    + <SLOT NAME = "Styles" >
  + <SLOT NAME = "metasign" >
  + <SLOT NAME = "listattachproc" >
```

****

```
- <FRAME NAME = "FrameTS_13.xml" >
    <NAMETERM VALUE = "регулируется" >
    <SLOT NAME = "NameCO" VALUE = "Действие" >
    <SLOT NAME = "KindAct" VALUE = "функция" >
  + <SLOT NAME = "Definitions" >
  + <SLOT NAME = "Propeties" >
  - <SLOT NAME = "SubjectObject" >
    - <SLOT NAME = "SO" >
        <SLOT NAME = "TypeSO" VALUE = "субъект" >
        <SLOT NAME = "NameTerm" VALUE = "трудовой договор" >
        <SLOT NAME = "Link" FILE = "FrameTS_45.xml" >
      <SLOT NAME = "AttachProc" VALUE = "FunSO" >
  + <SLOT NAME = "Tools" >
  + <SLOT NAME = "KvantRel" >
  + <SLOT NAME = "Actions" >
  + <SLOT NAME = "Events" >
  + <SLOT NAME = "MetaSign" >
  + <SLOT NAME = "ListAttachProc" >
```

****

```
<FRAME NAME = "FrameTS_19.xml" >
    <NAMETERM VALUE = "Сдельная заработная плата" >
    <SLOT NAME = "NameCO" VALUE = "Понятие" >
    <SLOT NAME = "KindEntity" VALUE = "Материальный" >
    <SLOT NAME = "definitions" >
        <SLOT NAME = "definition" VALUE = "денежное вознаграждение, выплачиваемое работодателем пропорционально производительности труда
        <SLOT NAME = "attachproc" VALUE = "FunDef" >
    <SLOT NAME = "propeties" >
    <SLOT NAME = "actions" >
    <SLOT NAME = "kvantrel" >
    <SLOT NAME = "states" >
    <SLOT NAME = "kvalitrel" >
        <SLOT NAME = "ClassKind" >
            <SLOT NAME = "Class" >
                <SLOT NAME = "nameclass" VALUE = "заработная плата" >
                <SLOT NAME = "link" FILE = "FrameTS_10.xml" >
                <SLOT NAME = "attachproc" VALUE = "FunClass" >
            <SLOT NAME = "Kind" >
        <SLOT NAME = "WholePart" >
        <SLOT NAME = "PresMethods" >
        <SLOT NAME = "Styles" >
    <SLOT NAME = "metasign" >
    <SLOT NAME = "listattachproc" >
```

****

```
<FRAME NAME = "FrameTS_20.xml" >
    <NAMETERM VALUE = "Повременная заработная плата" >
    <SLOT NAME = "NameCO" VALUE = "Понятие" >
    <SLOT NAME = "KindEntity" VALUE = "Материальный" >
    <SLOT NAME = "definitions" >
        <SLOT NAME = "definition" VALUE = "денежное вознаграждение, выплачиваемое работодателем пропорционально отработанному времени" >
        <SLOT NAME = "attachproc" VALUE = "FunDef" >
    <SLOT NAME = "propeties" >
    <SLOT NAME = "actions" >
    <SLOT NAME = "kvantrel" >
    <SLOT NAME = "states" >
    <SLOT NAME = "kvalitrel" >
        <SLOT NAME = "ClassKind" >
            <SLOT NAME = "Class" >
                <SLOT NAME = "nameclass" VALUE = "заработная плата" >
                <SLOT NAME = "link" FILE = "FrameTS_10.xml" >
                <SLOT NAME = "attachproc" VALUE = "FunClass" >
            <SLOT NAME = "Kind" >
        <SLOT NAME = "WholePart" >
        <SLOT NAME = "PresMethods" >
        <SLOT NAME = "Styles" >
    <SLOT NAME = "metasign" >
    <SLOT NAME = "listattachproc" >
```

2. Примеры словарных статей номенклатуры

```
        <SLOT NAME = "Kind" >
            <SLOT NAME = "namekind" VALUE = "реальная заработная плата" >
            <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
            <SLOT NAME = "attachproc" VALUE = "FunKind" >
    <SLOT NAME = "WholePart" >
        <SLOT NAME = "Whole" >
        <SLOT NAME = "Part" >
            <SLOT NAME = "namepart" VALUE = "регулярная премия" >
            <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
            <SLOT NAME = "attachproc" VALUE = "FunPart" >
        <SLOT NAME = "Part" >
            <SLOT NAME = "namepart" VALUE = "тринадцатая зарплата" >
            <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
            <SLOT NAME = "attachproc" VALUE = "FunPart" >
        <SLOT NAME = "Part" >
            <SLOT NAME = "namepart" VALUE = "поощрительная выплата" >
            <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
            <SLOT NAME = "attachproc" VALUE = "FunPart" >
        <SLOT NAME = "Part" >
            <SLOT NAME = "namepart" VALUE = "пособие по болезни" >
            <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
            <SLOT NAME = "attachproc" VALUE = "FunPart" >
        <SLOT NAME = "Part" >
            <SLOT NAME = "namepart" VALUE = "выходное пособие" >
            <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
            <SLOT NAME = "attachproc" VALUE = "FunPart" >
        <SLOT NAME = "PresMethods" >
        <SLOT NAME = "Styles" >
    <SLOT NAME = "metasign" >
    <SLOT NAME = "listattachproc" >
```

****

```
<FRAME NAME = "FrameTS_101.xml" >
    <NAMETERM VALUE = "размер минимальной заработной платы" >
    <SLOT NAME = "NameCO" VALUE = "Величина" >
    <SLOT NAME = "TypeQuantity" VALUE = "цена" >
    <SLOT NAME = "definitions" >
        <SLOT NAME = "definition" VALUE = "размер заработной платы, которая обеспечивает прожиточный минимум населения и соответ
        <SLOT NAME = "attachproc" VALUE = "FunDef" >
    <SLOT NAME = "MeasurementScale" >
        <SLOT NAME = "QualitativeMS" >
            <SLOT NAME = "SetValue1" VALUE = "прожиточный минимум трудоспособного человека" >
            <SLOT NAME = "AttachProc" VALUE = "FunQualitMS " >
        <SLOT NAME = "QuantitativeMS" >
        <SLOT NAME = "AttachProc" VALUE = "FunQuantitMS " >
    <SLOT NAME = "InterOfValues" >
    <SLOT NAME = "UnitMeasure" VALUE = "рубль" >
    <SLOT NAME = "MetaSign" >
    <SLOT NAME = "ListAttachProc" >
```

****

```
<FRAME NAME = "FrameNS_102.xml" >
    <NAMETERM VALUE = "сдельная заработная плата" >
    <SLOT NAME = "NameCO" VALUE = "Понятие" >
    <SLOT NAME = "KindEntity" VALUE = "Материальный" >
  <SLOT NAME = "definitions" >
      <SLOT NAME = "definition" VALUE = "это форма оплаты труда наёмного работника, при которой заработок зависит от количества произве
      <SLOT NAME = "attachproc" VALUE = "FunDef" >
  <SLOT NAME = "propeties" >
    <SLOT NAME = "property" >
        <SLOT NAME = "nameprop" VALUE = "улучшение количественных показателей работы" >
        <SLOT NAME = "link" FILE = "FrameNS_30.xml" >
    <SLOT NAME = "property" >
        <SLOT NAME = "nameprop" VALUE = "снижение качества производимой продукции" >
        <SLOT NAME = "link" FILE = "FrameNS_31.xml" >
    <SLOT NAME = "property" >
        <SLOT NAME = "nameprop" VALUE = "ухудшение обслуживания оборудования и как результат преждевременного выхода оборудовани
        <SLOT NAME = "link" FILE = "FrameNS_32.xml" >
    <SLOT NAME = "property" >
        <SLOT NAME = "nameprop" VALUE = "нарушение режима технологического процесса" >
        <SLOT NAME = "link" FILE = "FrameNS_33.xml" >
    <SLOT NAME = "property" >
        <SLOT NAME = "nameprop" VALUE = "нарушение требований техники безопасности" >
        <SLOT NAME = "link" FILE = "FrameNS_34.xml" >
    <SLOT NAME = "property" >
        <SLOT NAME = "nameprop" VALUE = "перерасход сырья и материалов" >
        <SLOT NAME = "link" FILE = "FrameNS_35.xml" >
      <SLOT NAME = "attachproc" VALUE = "FunProp" >
  <SLOT NAME = "actions" >
  <SLOT NAME = "kvantrel" >
  <SLOT NAME = "states" >
  <SLOT NAME = "kvalitrel" >
    <SLOT NAME = "ClassKind" >
      <SLOT NAME = "Class" >
          <SLOT NAME = "nameclass" VALUE = "заработная плата" >
          <SLOT NAME = "link" FILE = "FrameNS_100.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunClass" >
      <SLOT NAME = "Kind" >
          <SLOT NAME = "namekind" VALUE = "Прямая сдельная заработная плата" >
          <SLOT NAME = "link" FILE = "FrameNS_1020.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
          <SLOT NAME = "namekind" VALUE = "Сдельно-премиальная заработная плата" >
          <SLOT NAME = "link" FILE = "FrameNS_1021.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
          <SLOT NAME = "namekind" VALUE = "Косвенно-сдельная заработная плата" >
          <SLOT NAME = "link" FILE = "FrameNS_1022.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
          <SLOT NAME = "namekind" VALUE = "Аккордная сдельная заработная плата" >
          <SLOT NAME = "link" FILE = "FrameNS_1023.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
          <SLOT NAME = "namekind" VALUE = "Сдельно-прогрессивная заработная плата" >
          <SLOT NAME = "link" FILE = "FrameNS_1024.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
          <SLOT NAME = "namekind" VALUE = "Смешанная (повременно-сдельная) заработная плата" >
          <SLOT NAME = "link" FILE = "FrameNS_1025.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunKind" >
    <SLOT NAME = "WholePart" >
      <SLOT NAME = "Whole" >
      <SLOT NAME = "Part" >
          <SLOT NAME = "namepart" VALUE = "сдельные расценки" >
          <SLOT NAME = "link" FILE = "FrameNS_156.xml" >
          <SLOT NAME = "attachproc" VALUE = "FunPart" >
    <SLOT NAME = "PresMethods" >
    <SLOT NAME = "Styles" >
  <SLOT NAME = "metasign" >
  <SLOT NAME = "listattachproc" >
```

****

```
<FRAME NAME = "FrameNS_103.xml" >
  <NAMETERM VALUE = "Повременная заработная плата" >
  <SLOT NAME = "NameCO" VALUE = "Понятие" >
  <SLOT NAME = "KindEntity" VALUE = "Материальный" >
  <SLOT NAME = "definitions" >
    <SLOT NAME = "definition" VALUE = "это форма оплаты труда наёмного работника, при которой заработок зависит от количества фактиче
    <SLOT NAME = "attachproc" VALUE = "FunDef" >
  <SLOT NAME = "propeties" >
    <SLOT NAME = "property" >
      <SLOT NAME = "nameprop" VALUE = "повышение квалификации работников и укрепление дисциплины труда" >
      <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
    <SLOT NAME = "attachproc" VALUE = "FunProp" >
  <SLOT NAME = "actions" >
  <SLOT NAME = "kvantrel" >
  <SLOT NAME = "states" >
  <SLOT NAME = "kvalitrel" >
    <SLOT NAME = "ClassKind" >
      <SLOT NAME = "Class" >
        <SLOT NAME = "nameclass" VALUE = "заработная плата" >
        <SLOT NAME = "link" FILE = "FrameNS_100.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunClass" >
      <SLOT NAME = "Kind" >
        <SLOT NAME = "namekind" VALUE = "Простая повременная" >
        <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
        <SLOT NAME = "namekind" VALUE = "Повременно-премиальная" >
        <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
        <SLOT NAME = "namekind" VALUE = "Повременная с нормированным заданием" >
        <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunKind" >
      <SLOT NAME = "Kind" >
        <SLOT NAME = "namekind" VALUE = "Смешанная (повременно-сдельная)" >
        <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
        <SLOT NAME = "attachproc" VALUE = "FunKind" >
    <SLOT NAME = "WholePart" >
      <SLOT NAME = "Whole" >
      <SLOT NAME = "Part" >
    <SLOT NAME = "PresMethods" >
    <SLOT NAME = "Styles" >
      <SLOT NAME = "Style" >
        <SLOT NAME = "namemeth" VALUE = "null" >
        <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
      <SLOT NAME = "attachproc" VALUE = "FunStyle" >
  <SLOT NAME = "metasign" >
    <SLOT NAME = "MetalangPresMethods" >
      <SLOT NAME = "MetalangPresMeth" >
        <SLOT NAME = "namemeth" VALUE = "null" >
        <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
      <SLOT NAME = "attachproc" VALUE = "FunMetalangPres" >
    <SLOT NAME = "TermOthLang" >
      <SLOT NAME = "TermOthLang" >
        <SLOT NAME = "namemeth" VALUE = "time-rate pay" >
        <SLOT NAME = "link" FILE = "FrameNS_#.xml" >
      <SLOT NAME = "attachproc" VALUE = "FunTOthLang" >
  <SLOT NAME = "listattachproc" >
```

ПРИЛОЖЕНИЕ Г

Пример XML-описания спецификации метода «Морфологический анализ»

```
<METHOD NAME = "Method_MA" >
  <RESEARCH_OBJECT> Лексема </RESEARCH_OBJECT>
  <SET_CONSTRUCTS>
    <CONSTRUCT NUMBER = "1" >
      <ARC TERM = "Семантическое отношение" VALUE = "принадлежит" >
      <VERTEX TYPE = "left" NAMESET = "A" VALUE = "Лексема" >
      <VERTEX TYPE = "right" NAMESET = "B" VALUE = "словарьСГС" >
    <CONSTRUCT NUMBER = "2" >
      <ARC TERM = "Семантическое отношение" VALUE = "принадлежит" >
      <VERTEX TYPE = "left" NAMESET = "A" VALUE = "Основа" >
      <VERTEX TYPE = "right" NAMESET = "B" VALUE = "словарьОснов" >
    <CONSTRUCT NUMBER = "3" >
      <ARC TERM = "Семантическое отношение" VALUE = "принадлежит" >
      <VERTEX TYPE = "left" NAMESET = "A" VALUE = "Окончание" >
      <VERTEX TYPE = "right" NAMESET = "B" VALUE = "словарьОкончаний" >
    <CONSTRUCT NUMBER = "4" >
      <ARC TERM = "Семантическое отношение" VALUE = "содержит" >
      <VERTEX TYPE = "left" NAMESET = "C" VALUE = "Лексема" >
      <VERTEX TYPE = "right" NAMESET = "D" VALUE = "Основа" >
      <VERTEX TYPE = "right" NAMESET = "D" VALUE = "Окончание" >
    <CONSTRUCT NUMBER = "5" >
      <ARC TERM = "Семантическое отношение" VALUE = "содержит" >
      <VERTEX TYPE = "left" NAMESET = "C" VALUE = "СГС" >
      <VERTEX TYPE = "left" NAMESET = "C" VALUE = "СОС" >
      <VERTEX TYPE = "left" NAMESET = "C" VALUE = "СОК" >
      <VERTEX TYPE = "right" NAMESET = "D" VALUE = "Вектор" >
    <CONSTRUCT NUMBER = "6" >
      <ARC TERM = "Семантическое отношение" VALUE = "содержит" >
      <VERTEX TYPE = "left" NAMESET = "C" VALUE = "Вектор" >
      <VERTEX TYPE = "right" NAMESET = "D" VALUE = "Компонент" >
    <CONSTRUCT NUMBER = "7" >
      <ARC TERM = "Семантическое отношение" VALUE = "имеет" >
      <VERTEX TYPE = "left" NAMESET = "Словарь" VALUE = "Словарь" >
      <VERTEX TYPE = "right" NAMESET = "Характеристика" VALUE = "Характеристика" >
    <CONSTRUCT NUMBER = "8" >
      <ARC TERM = "Семантическое отношение" VALUE = "есть" >
      <VERTEX TYPE = "left" NAMESET = "Характеристика" VALUE = "Характеристика" >
      <VERTEX TYPE = "right" NAMESET = "Имя" VALUE = "Имя" >
    <CONSTRUCT NUMBER = "9" >
      <ARC TERM = "Семантическое отношение" VALUE = "имеет" >
      <VERTEX TYPE = "left" NAMESET = "Имя" VALUE = "Имя" >
      <VERTEX TYPE = "right" NAMESET = "Значение" VALUE = "Значение" >
    <CONSTRUCT NUMBER = "10" >
      <ARC TERM = "Семантическое отношение" VALUE = "есть" >
      <VERTEX TYPE = "left" NAMESET = "Компонент" VALUE = "Компонент" >
      <VERTEX TYPE = "right" NAMESET = "E" VALUE = "ГотСловоформа" >
      <VERTEX TYPE = "right" NAMESET = "E" VALUE = "ЧастьРечи" >
      <VERTEX TYPE = "right" NAMESET = "E" VALUE = "Основа" >
      <VERTEX TYPE = "right" NAMESET = "E" VALUE = "Окончание" >
    <CONSTRUCT NUMBER = "11" >
      <ARC TERM = "Семантическое отношение" VALUE = "эквивалентен" >
      <VERTEX TYPE = "left" NAMESET = "A" VALUE = "Лексема" >
      <VERTEX TYPE = "right" NAMESET = "K" VALUE = "ГотСловоформа" >
    <CONSTRUCT NUMBER = "12" >
      <ARC TERM = "Семантическое отношение" VALUE = "эквивалентен" >
      <VERTEX TYPE = "left" NAMESET = "A" VALUE = "Основа" >
      <VERTEX TYPE = "right" NAMESET = "K" VALUE = "Основа" >
```

```
<LEVEL_1_TREE>
  <LIST_VERTEX NUMBER_VARIANT = "1" >
    <ROOT_BRANCH CODE_ROOT = "v01" > словарь </ROOT_BRANCH>
  <LIST_VERTEX NUMBER_VARIANT = "2" >
    <ROOT_BRANCH CODE_ROOT = "v02" > Основа </ROOT_BRANCH>
    <ROOT_BRANCH CODE_ROOT = "v03" > Окончание </ROOT_BRANCH>
  <LIST_VERTEX NUMBER_VARIANT = "3" >
    <ROOT_BRANCH CODE_ROOT = "v02" > Основа </ROOT_BRANCH>
  <LIST_VERTEX NUMBER_VARIANT = "4" >
    <ROOT_BRANCH CODE_ROOT = "v01" > Словарь </ROOT_BRANCH>
    <ROOT_BRANCH CODE_ROOT = "v04" > Характеристика </ROOT_BRANCH>
    <ROOT_BRANCH CODE_ROOT = "v04" > Характеристика </ROOT_BRANCH>
  <LIST_VERTEX NUMBER_VARIANT = "5" >
    <ROOT_BRANCH CODE_ROOT = "v03" > Окончание </ROOT_BRANCH>
    <ROOT_BRANCH CODE_ROOT = "v04" > Характеристика </ROOT_BRANCH>
    <ROOT_BRANCH CODE_ROOT = "v04" > Характеристика </ROOT_BRANCH>
<SET_BRANCH_TREE>
  <BRANCH_TREE CODE_ROOT = "v01" BRANCH_ROOT = "v01" >
    <ROOT VERTEX = "Словарь" >
      <ROOT VERTEX = "Характеристика" >
        <ROOT VERTEX = "Имя" >
          <ROOT VERTEX = "Значение" >
      <ROOT VERTEX = "Вектор_RWord" >
        <ROOT VERTEX = "Компонент" >
          <ROOT VERTEX = "ЧастьРечи" >
            <ROOT VERTEX = "Значение" >
        <ROOT VERTEX = "Компонент" >
          <ROOT VERTEX = "ГотСловоформа" >
            <ROOT VERTEX = "лексема" >
    <ROOT VERTEX = "Словарь" >
      <ROOT VERTEX = "Характеристика" >
        <ROOT VERTEX = "Имя" >
          <ROOT VERTEX = "Значение" >
      <ROOT VERTEX = "Вектор_SMI" >
        <ROOT VERTEX = "Компонент" >
          <ROOT VERTEX = "КодФК" >
            <ROOT VERTEX = "КодФК" >
        <ROOT VERTEX = "Компонент" >
          <ROOT VERTEX = "ЧастьРечи" >
            <ROOT VERTEX = "ЧастьРечи" >
        <ROOT VERTEX = "Компонент" >
          <ROOT VERTEX = "Род" >
        <ROOT VERTEX = "Компонент" >
          <ROOT VERTEX = "ПризнакОдушевленности" >
  <BRANCH_TREE CODE_ROOT = "v02" BRANCH_ROOT = "v02" >
    <ROOT VERTEX = "Основа" >
      <ROOT VERTEX = "Словарь" >
        <ROOT VERTEX = "Характеристика" >
          <ROOT VERTEX = "Имя" >
            <ROOT VERTEX = "Значение" >
        <ROOT VERTEX = "Вектор_Base" >
          <ROOT VERTEX = "Компонент" >
            <ROOT VERTEX = "ЧастьРечи" >
              <ROOT VERTEX = "Значение" >
              <ROOT VERTEX = "ЧастьРечи" >
          <ROOT VERTEX = "Компонент" >
            <ROOT VERTEX = "Основа" >
              <ROOT VERTEX = "Основа" >
```

```
<ROOT VERTEX = "Основа" >
<ROOT VERTEX = "Словарь" >
<ROOT VERTEX = "Характеристика" >
<ROOT VERTEX = "Имя" >
<ROOT VERTEX = "Значение" >
<ROOT VERTEX = "Вектор_FK" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "КодФК" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "Основа" >
<ROOT VERTEX = "Основа" >
<BRANCH_TREE CODE_ROOT = "v03" BRANCH_ROOT = "v03" >
<ROOT VERTEX = "Окончание" >
<ROOT VERTEX = "Словарь" >
<ROOT VERTEX = "Характеристика" >
<ROOT VERTEX = "Имя" >
<ROOT VERTEX = "Значение" >
<ROOT VERTEX = "Вектор_End" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "ЧастьРечи" >
<ROOT VERTEX = "Значение" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "Окончание" >
<ROOT VERTEX = "Окончание" >
<ROOT VERTEX = "Окончание" >
<ROOT VERTEX = "Словарь" >
<ROOT VERTEX = "Характеристика" >
<ROOT VERTEX = "Имя" >
<ROOT VERTEX = "Значение" >
<ROOT VERTEX = "Вектор_DMI" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "КодФК" >
<ROOT VERTEX = "КодФК" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "ЧастьРечи" >
<ROOT VERTEX = "ЧастьРечи" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "Окончание" >
<ROOT VERTEX = "Окончание" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "Падеж" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "Число" >
<ROOT VERTEX = "Компонент" >
<ROOT VERTEX = "Лицо" >
<BRANCH_TREE CODE_ROOT = "v04" BRANCH_ROOT = "v04" >
<ROOT VERTEX = "Характеристика" >
<ROOT VERTEX = "Имя" >
<ROOT VERTEX = "Характеристика" >
<ROOT VERTEX = "ЧастьРечи" >
<ROOT VERTEX = "Характеристика" >
<ROOT VERTEX = "КодФК" >
<ROOT VERTEX = "Характеристика" >
<ROOT VERTEX = "Имя" >
<ROOT VERTEX = "Значение" >
<SET_VALUE_VERTEX>
<LIST_TERM NAME = "A" >
<TERM VALUE = "Лексема" SIGN = "l" >
<TERM VALUE = "Основа" SIGN = "b" >
<TERM VALUE = "Окончание" SIGN = "o" >
```

```
<LIST_TERM NAME = "B" >
    <TERM VALUE = "Словарь готовых словоформ (СГС)" SIGN = "d" >
    <TERM VALUE = "Словарь основ (СОС)" SIGN = "d" >
    <TERM VALUE = "Словарь окончаний (СОК)" SIGN = "d" >
<LIST_TERM NAME = "C" >
    <TERM VALUE = "Лексема" SIGN = "l" >
    <TERM VALUE = "Словарь готовых словоформ (СГС)" SIGN = "d" >
    <TERM VALUE = "Словарь основ (СОС)" SIGN = "d" >
    <TERM VALUE = "Словарь окончаний (СОК)" SIGN = "d" >
<LIST_TERM NAME = "D" >
    <TERM VALUE = "Основа" SIGN = "b" >
    <TERM VALUE = "Окончание" SIGN = "o" >
    <TERM VALUE = "Вектор" SIGN = "v_" >
    <TERM VALUE = "Компонент" SIGN = "g" >
<LIST_TERM NAME = "E" >
    <TERM VALUE = "ГотСловоформа" SIGN = "c" >
    <TERM VALUE = "ЧастьРечи" SIGN = "c" >
    <TERM VALUE = "Основа" SIGN = "b" >
    <TERM VALUE = "Окончание" SIGN = "o" >
<LIST_TERM NAME = "K" >
    <TERM VALUE = "ГотСловоформа" SIGN = "c" >
    <TERM VALUE = "Основа" SIGN = "b" >
    <TERM VALUE = "Окончание" SIGN = "o" >
<VALUE_TERM VALUE = "Словарь" >
    <VALUE>СГС</VALUE>
    <VALUE>СОС</VALUE>
    <VALUE>СОК</VALUE>
    <VALUE>Словарь ФК</VALUE>
    <VALUE>Словарь СМИ</VALUE>
    <VALUE>Словарь ДМИ</VALUE>
<VALUE_TERM VALUE = "ЧастьРечи" >
    <VALUE>Сущ</VALUE>
    <VALUE>Глагол</VALUE>
    <VALUE>Прил</VALUE>
    <VALUE>Причастие</VALUE>
    <VALUE>Числительное</VALUE>
    <VALUE>Союз</VALUE>
    <VALUE>Местоимение</VALUE>
    <VALUE>Деепричастие</VALUE>
    <VALUE>Предлог</VALUE>
    <VALUE>Частица</VALUE>
    <VALUE>Вводное слово</VALUE>
```

ПРИЛОЖЕНИЕ Д

Пример файла “ConfigTask.xml”

Файл “ConfigTask.xml”, содержащий спецификацию данных о предметной области задачи

```
<TASK>
  <ALPHABET>
    <ALPHALIST>
      <SYMBOL NAME = "y.term" >
      <SYMBOL NAME = "y.sign" >
      <SYMBOL NAME = "x.term" >
      <SYMBOL NAME = "x.sign" >
      <SYMBOL NAME = "r.semrel" >
      <SYMBOL NAME = "operator." >
      <SYMBOL NAME = "operator.eof" >
    <BETALIST>
      <SYMBOL NAME = "Valueterm" >
      <SYMBOL NAME = "Valuesign" >
      <SYMBOL NAME = "Valueop" >
      <SYMBOL NAME = "Valuesemrel" >
  <STATE_ACT>
    <STATELIST>
      <SYMBOL NAME = "занесение beta в стек StYX" >
      <SYMBOL NAME = "занесение beta в стек StR" >
      <SYMBOL NAME = "занесение beta в стек StVT" >
      <SYMBOL NAME = "чтение входного потока" >
      <SYMBOL NAME = "распознавание отношения R" >
      <SYMBOL NAME = "занесение beta в стек StOp beta" >
      <SYMBOL NAME = "пустое действие" >
      <SYMBOL NAME = "формирование выходного XML-дерева" >
    <ACTLIST>
      <SYMBOL NAME = "занесение beta в стек StYX" >
      <SYMBOL NAME = "занесение beta в стек StR" >
      <SYMBOL NAME = "занесение beta в стек StVT" >
      <SYMBOL NAME = "чтение входного потока" >
      <SYMBOL NAME = "распознавание отношения R" >
      <SYMBOL NAME = "занесение beta в стек StOp beta" >
      <SYMBOL NAME = "пустое действие" >
      <SYMBOL NAME = "формирование выходного XML-дерева" >
  <LIST_FILE>
    <ETALONNAME>
      <FILE NAME = "Etalon1.xml" >
    <INPUTNAME>
      <FILE NAME = "Input1.xml" >
    <OUTPUTNAME>
      <FILE NAME = "Output1.xml" >
  <PARAMETRES_GA>
    <CICLECOUNT> 1000 </CICLECOUNT>
    <DELTA>0.0 </DELTA>
    <POPSIZE> 2000 </POPSIZE>
    <PARPOOL> 0.1 </PARPOOL>
    <MUTPOOL> 0.3 </MUTPOOL>
    <TOURNUM> 10 </TOURNUM>
    <ROULNUM> 4 </ROULNUM>
    <SELTYPE> 0 </SELTYPE>
  <LIBRARY_LIST>
    <STANDARTLIB>
      <LIBRARY NAME = "?" >
      <LIBRARY NAME = "?" >
    <USERLIB>
      <LIBRARY NAME = "?" >
      <LIBRARY NAME = "?" >
  <PARAMETRES_SA>
    <ARITY>3 </ARITY>
    <TYPEOA> отношение "Целое-Часть" </TYPEOA>
```

ПРИЛОЖЕНИЕ Е

Пример XML-документа «Input.xml»

Входная продукция

```
<INPUT>
  <SYMBOL TYPE = "operator" VALUE = "if">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "КомпСловоСоч">
    <SYMBOL TYPE = "sign" VALUE = "k1">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "предложение">
    <SYMBOL TYPE = "sign" VALUE = "p1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "содержит">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "ИССловоСоч">
    <SYMBOL TYPE = "sign" VALUE = "s1">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "КомпСловоСоч">
    <SYMBOL TYPE = "sign" VALUE = "k1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "содержит">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "ИССловоСоч">
    <SYMBOL TYPE = "sign" VALUE = "s2">
  <SYMBOL TYPE = "x">
    <symbol type = "term" VALUE = "КомпСловоСоч">
    <SYMBOL TYPE = "sign" VALUE = "k1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "содержит">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "терм-спутникX">
    <SYMBOL TYPE = "sign" VALUE = "tx1">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "ИССловоСоч">
    <SYMBOL TYPE = "sign" VALUE = "s1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "содержит">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "значение">
    <SYMBOL TYPE = "sign" VALUE = "часть">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "терм-спутникX">
    <SYMBOL TYPE = "sign" VALUE = "tx1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "имеет">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "термин">
    <SYMBOL TYPE = "sign" VALUE = "z1">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "ИССловоСоч">
    <SYMBOL TYPE = "sign" VALUE = "s2">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "содержит">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "характеристика">
    <SYMBOL TYPE = "sign" VALUE = "x1">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "термин">
    <SYMBOL TYPE = "sign" VALUE = "z1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "имеет">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "падеж">
    <SYMBOL TYPE = "sign" VALUE = "c1">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "характеристика">
    <SYMBOL TYPE = "sign" VALUE = "x1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "есть">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "значение">
    <SYMBOL TYPE = "sign" VALUE = "Род">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "падеж">
    <SYMBOL TYPE = "sign" VALUE = "c1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "имеет">
  <SYMBOL TYPE = "operator" VALUE = "then">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "тип">
    <SYMBOL TYPE = "sign" VALUE = "Часть">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "термин">
    <SYMBOL TYPE = "sign" VALUE = "z1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "имеет">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "тип">
    <SYMBOL TYPE = "sign" VALUE = "Целое">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "ИССловоСоч">
    <SYMBOL TYPE = "sign" VALUE = "s2">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "имеет">
  <SYMBOL TYPE = "operator" VALUE = "and">
  <SYMBOL TYPE = "y">
    <SYMBOL TYPE = "term" VALUE = "категория">
    <SYMBOL TYPE = "sign" VALUE = "ЦелоеЧасть">
  <SYMBOL TYPE = "x">
    <SYMBOL TYPE = "term" VALUE = "СемОтношение">
    <SYMBOL TYPE = "sign" VALUE = "r1">
  <SYMBOL TYPE = "r">
    <SYMBOL TYPE = "semrel" VALUE = "относится_к">
  <SYMBOL TYPE = "operator" VALUE = "eof">
```

ПРИЛОЖЕНИЕ Ж

Пример XML-документа «Etalon.xml»

Эталонная продукция

```
<ETALON>
  <SYMBOL TYPE = "operator" VALUE = "if" >
  <SYMBOL TYPE = "PS" VALUE = "PAggr" >
    <SYMBOL TYPE = "FArg" VALUE = "whole" >
    <SYMBOL TYPE = "X" VALUE = "p1" >
    <SYMBOL TYPE = "Y" VALUE = "k1" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PAggr" >
    <SYMBOL TYPE = "FArg" VALUE = "whole" >
    <SYMBOL TYPE = "X" VALUE = "k1" >
    <SYMBOL TYPE = "Y" VALUE = "s1" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PAggr" >
    <SYMBOL TYPE = "FArg" VALUE = "whole" >
    <SYMBOL TYPE = "X" VALUE = "k1" >
    <SYMBOL TYPE = "Y" VALUE = "s2" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PAggr" >
    <SYMBOL TYPE = "FArg" VALUE = "whole" >
    <SYMBOL TYPE = "X" VALUE = "s1" >
    <SYMBOL TYPE = "Y" VALUE = "tx1" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PHier" >
    <SYMBOL TYPE = "FArg" VALUE = "value" >
    <SYMBOL TYPE = "X" VALUE = "tx1" >
    <SYMBOL TYPE = "Y" VALUE = "часть" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PAggr" >
    <SYMBOL TYPE = "FArg" VALUE = "whole" >
    <SYMBOL TYPE = "X" VALUE = "s2" >
    <SYMBOL TYPE = "Y" VALUE = "z1" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PProp" >
    <SYMBOL TYPE = "FArg" VALUE = "character" >
    <SYMBOL TYPE = "X" VALUE = "z1" >
    <SYMBOL TYPE = "Y" VALUE = "x1" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PHier" >
    <SYMBOL TYPE = "FArg" VALUE = "category" >
    <SYMBOL TYPE = "X" VALUE = "x1" >
    <SYMBOL TYPE = "Y" VALUE = "c1" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PHier" >
    <SYMBOL TYPE = "FArg" VALUE = "value" >
    <SYMBOL TYPE = "X" VALUE = "c1" >
    <SYMBOL TYPE = "Y" VALUE = "Род" >
  <SYMBOL TYPE = "operator" VALUE = "then" >
  <SYMBOL TYPE = "PS" VALUE = "PHier" >
    <SYMBOL TYPE = "FArg" VALUE = "category" >
    <SYMBOL TYPE = "X" VALUE = "z1" >
    <SYMBOL TYPE = "Y" VALUE = "Часть" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PHier" >
    <SYMBOL TYPE = "FArg" VALUE = "category" >
    <SYMBOL TYPE = "X" VALUE = "s2" >
    <SYMBOL TYPE = "Y" VALUE = "Целое" >
  <SYMBOL TYPE = "operator" VALUE = "and" >
  <SYMBOL TYPE = "PS" VALUE = "PHier" >
    <SYMBOL TYPE = "FArg" VALUE = "category" >
    <SYMBOL TYPE = "X" VALUE = "r1" >
    <SYMBOL TYPE = "Y" VALUE = "ЦелоеЧасть" >
  <SYMBOL TYPE = "operator" VALUE = "eof" >
```

Найханова Лариса Владимировна

ТЕХНОЛОГИЯ СОЗДАНИЯ МЕТОДОВ АВТОМАТИЧЕСКОГО ПОСТРОЕНИЯ ОНТОЛОГИЙ С ПРИМЕНЕНИЕМ ГЕНЕТИЧЕСКОГО И АВТОМАТНОГО ПРОГРАММИРОВАНИЯ

Научное издание

Печатается по решению Ученого Совета Восточно-Сибирского государственного технологического университета от 28.08.2008 г.

Редактор *Т.А. Стороженко*

Подписано в печать 15.09.2008 г. Формат 60×84 1/16.
Печать операт., бумага писч. Усл. печ. л. 14,18.
Тираж 150 экз. Заказ № 189.

Редакционно-издательский отдел
Издательства БНЦ СО РАН
670047 г. Улан-Удэ, ул. Сахьяновой, 8.

Отпечатано в типографии ВСГТУ
670013 г. Улан-Удэ, ул. Ключевская, 40 в.