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

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

ТЕХНОЛОГИИ И ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 6-05-0611-03 «Искусственный интеллект»

Минск БГУИР 2026

УДК 004.8(076)
ББК 32.813я73
Т38

Авторы:

Н. В. Гракова, М. Е. Садовский, А. В. Жмырко, Н. В. Зотов

Рецензенты:

кафедра интеллектуальных систем
Белорусского государственного университета
(протокол № 8 от 10.03.2025);

заведующий кафедрой технологий программирования
учреждения образования «Полоцкий государственный университет
имени Евфросинии Полоцкой» кандидат технических наук,
доцент В. М. Чертков

Технологии и инструментальные средства проектирования
Т38 интеллектуальных систем : учеб.-метод. пособие / Н. В. Гракова, М. Е. Садовский, А. В. Жмырко, Н. В. Зотов. – Минск : БГУИР, 2026. – 150 с.

ISBN 978-985-543-861-9.

Излагается теоретический материал с примерами решения практических задач. Кроме того, представлен материал для выполнения практических заданий и лабораторных работ по учебной дисциплине «Технологии и инструментальные средства проектирования интеллектуальных систем».

УДК 004.8(076)
ББК 32.813я73

ISBN 978-985-543-861-9

Содержание

Введение

Учебная дисциплина «Технологии и инструментальные средства проектирования интеллектуальных систем» является одной из числа общепрофессиональных и специальных дисциплин в подготовке студентов в области искусственного интеллекта. Ее назначение – изучение современных технологий проектирования семантически совместимых гибридных интеллектуальных компьютерных систем.

Цель преподавания учебной дисциплины: получение обучающимися теоретических представлений о методах и технологиях проектирования интеллектуальных систем, а также выработка практических навыков использования современных инструментальных средств для создания моделей интеллектуальных систем.

Задачи учебной дисциплины:

− приобретение знаний в области использования и классификации интеллектуальных систем;

− приобретение навыков о составе и организации знаний в интеллектуальных системах;

− приобретение умений по извлечению знаний из различных источников;

− изучение принципов построения интеллектуальных систем;

− овладение методами представления и структурирования знаний;

− овладение технологией проектирования систем, основанных на знаниях.

Данное учебно-методическое пособие содержит теоретический материал, необходимый для получения знаний о технологиях проектирования интеллектуальных компьютерных систем, а также средств проектирования таких систем. В рамках практических заданий обучающиеся смогут приобрести знания и навыки, связанные с инженерией знаний. Выполняя последовательно задания, предлагаемые в рамках лабораторных работ, с помощью технологии OSTIS (Open Semantic Technology for Intelligent Systems) и интеллектуальной обучающей экспертной системы NIKA (Nika is an Intelligent Knowledge-driven Assistant) учащиеся смогут освоить все этапы разработки интеллектуальных компьютерных систем.

1 Технологии проектирования интеллектуальных систем

Технология – комплекс моделей, методов и средств, обеспечивающих решение соответствующего класса задач, выполнение соответствующего класса действий, реализацию соответствующего вида деятельности. Технология проектирования чего-либо обычно сопровождается разработкой соответствующего стандарта проектирования. В рамках данного курса детальное внимание уделяется технологии проектирования интеллектуальных систем, а конкретно Технологии OSTIS [1, 2].

Технология OSTIS – это комплекс (семейство) технологий, обеспечивающих проектирование, производство, эксплуатацию и реинжиниринг интеллектуальных компьютерных систем (ostis-систем), которые предназначены для автоматизации самых различных видов человеческой деятельности, а в их основе лежит смысловое представление и онтологическая систематизация знаний и агентно-ориентированная обработка знаний [1]. Другими словами, Технология OSTIS – это технология принципиально нового уровня, целью и задачами которой являются:

− разработка интеллектуальных компьютерных систем (ostis-систем), семантически совместимых между собой, способных самостоятельно взаимодействовать, адаптироваться к пользователям и адаптировать (обучать) самих пользователей к более эффективному взаимодействию с интеллектуальными компьютерными системами по принципам, лежащим в основе этой технологии;

− интеграция и конвергенция самых различных видов знаний и самых различных моделей решения задач, неразрывно связанных с процессом разработки интеллектуальных компьютерных систем и процессом повышения квалификации разработчиков этих систем.

Разработка технологии проектирования интеллектуальных систем осуществляется едино по всем направлениям развития искусственного интеллекта.

Искусственный интеллект – научно-техническая деятельность, направленная на построение теории интеллектуальных систем, а также на создание технологии проектирования и производства искусственных

интеллектуальных систем (интеллектуальных компьютерных систем); область человеческой деятельности, основными целями которой являются:

− построение теории интеллектуальных систем;

− создание технологии разработки интеллектуальных компьютерных систем (искусственных интеллектуальных систем);

− переход на принципиально новый уровень комплексной автоматизации всех видов человеческой деятельности, который основан на массовом применении интеллектуальных компьютерных систем и предполагает не только наличие интеллектуальных компьютерных систем, способных понимать друг друга и согласовывать свою деятельность, но и построение общей теории человеческой деятельности, осуществляемой в условиях нового уровня ее автоматизации (теории деятельности smart-общества), которая должна быть понятна используемым интеллектуальным компьютерным системам и которая потребует существенного переосмысления современной организации человеческой деятельности.

Под любой ***интеллектуальной компьютерной системой*** (искусственной интеллектуальной системой) понимается искусственная кибернетическая система, обладающая высоким уровнем интеллекта (высоким уровнем знаний и умений), а также высоким уровнем обучаемости. Частным случаем таких систем являются ostis-системы, проектируемые на основе Технологии OSTIS.

Ostis-система [2] – интеллектуальная компьютерная система, которая построена в соответствии с требованиями и стандартами Технологии OSTIS, что обеспечивает существенное развитие целого ряда свойств (способностей) этой компьютерной системы, позволяющих значительно повысить уровень ее интеллекта (и, прежде всего, ее уровень обучаемости и уровень социализации); это значит, что обеспечены:

− семантическая совместимость (взаимопонимание) таких систем между собой;

− способность к их самостоятельному взаимодействию, к координации своей деятельности при возникновении непредсказуемых (нештатных) ситуаций.

Если интеллектуальные компьютерные системы не будут обладать указанными выше способностями, то ни о каких smart-предприятиях, smart-учреждениях, smart-городах, ни о каком smart-обществе и речи быть не может, так как все обстоятельства их деятельности заранее на этапе проектирования предусмотреть принципиально невозможно. Это означает, что интеллектуальные компьютерные системы должны научиться самостоятельно «отрабатывать» все заранее непредусмотренные обстоятельства.

В данном учебно-методическом пособии объектом изучения являются *интеллектуальные обучающие экспертные системы*. Каждая **интеллектуальная обучающая система** в качестве простейшего средства изучения учебного материала предполагает наличие средств навигации по этому материалу и средств задания по нему различных вопросов. Системы, обладающие только таким ограниченным набором возможностей, названы ***интеллектуальными справочными системами***. Таким образом, можно сказать, что интеллектуальная обучающая система обязательно реализует в себе функции интеллектуальной справочной системы. ***Интеллектуальная экспертная система*** – интеллектуальная система, включающая в себя знания об определенной слабо структурированной и трудно формализуемой узкой предметной области и способная предлагать и объяснять пользователю разумные решения.

Как упоминалось выше, важными процессами при проектировании и эксплуатации интеллектуальных компьютерных систем являются конвергенция и интеграция.

Конвергенция пар конкретных искусственных сущностей (например, технических систем) есть стремление к их унификации (в частности, к стандартизации), т. е. стремление к минимизации многообразия форм решения аналогичных практических задач – к тому, чтобы все, что можно сделать одинаково, сделалось одинаково, но без ущерба для требуемого качества. Последнее очень важно, так как безграмотная стандартизация может привести к существенному торможению прогресса. Ограничение многообразия форм не должно приводить к ограничению содержания, возможностей. Образно говоря, «словам должно быть тесно, а мыслям – свободно».

Часто под конвергенцией может пониматься конвергенция баз знаний, либо конвергенция моделей решения задач, либо конвергенция гибридных решателей задач.

Интеграция – объединение нескольких разных сущностей, в результате чего возникает некоторая объединенная целостная сущность.

Понятие интеграции имеет тесную связь с понятием конвергенции. Чем выше степень конвергенции (степень сближения) интегрируемых объектов, тем выше качество результата интеграции.

1.1 Информация, данные, знания, контент

При работе с информационными и/или интеллектуальными системами традиционно возникает вопрос о том, как соотносятся понятия «информация», «данные», «знания», «контент».

Информация (от лат. Informatio – «разъяснение, научение, сведение, изложение, осведомленность») – это сведения (данные), которые воспринимаются живым существом или устройством и сообщаются (получаются, передаются, преобразуются, сжимаются, теряются, находятся, регистрируются) с помощью знаков символьного, иконического, жесткого или звукового типа. Эти данные об окружающем мире (объектах, явлениях, событиях, процессах) уменьшают степень неопределенности людей, неполноту их знаний.

Данные – это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства [3].

При обработке на ЭВМ данные трансформируются, условно проходя следующие этапы [3]:

− **D1** – данные как результат измерений и наблюдений;

− **D2** – данные на материальных носителях информации (таблицы, протоколы, справочники);

− **D3** – модели (структуры) данных в виде диаграмм, графиков, функций;

− **D4** – данные в компьютере на языке описания данных;

− **D5** – базы данных на машинных носителях информации.

Знания – это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области [3].

Знания связаны с данными, основываются на них, но представляют результат мыслительной деятельности человека, обобщая его опыт, полученный в ходе выполнения какой-либо практической деятельности. Часто они получаются эмпирическим путем.

При обработке на ЭВМ знания трансформируются аналогично данным [3, 4]:

− **Z1** – знания в памяти человека как результат мышления;

− **Z2** – материальные носители знаний (учебники, методические пособия);

− **Z3** – **поле знаний** – условное описание основных объектов предметной области, их атрибутов и закономерностей, их связывающих;

− **Z4** – знания, описанные на языках представления знаний (продукционные языки, семантические сети, фреймы и др.);

− **Z5** – **база знаний на машинных носителях информации**.

Кроме того, для знания используют часто следующее определение. ***Знания*** – это хорошо структурированные данные, или данные о данных, или метаданные.

Существует множество способов определять понятия. Один из широко применяемых способов основан на идее интенсионала. ***Интенсионал понятия*** – это определение его через соотнесение с понятием более высокого уровня абстракции с указанием специфических свойств. Интенсионалы формулируют знания об объектах. Другой способ определяет понятие через соотнесение с понятиями более низкого уровня абстракции или перечисление фактов, относящихся к определяемому объекту. Это есть определение через данные, или ***экстенсионал понятия***.

Основные фазы обработки знаний представлены на рисунке 1 [4].

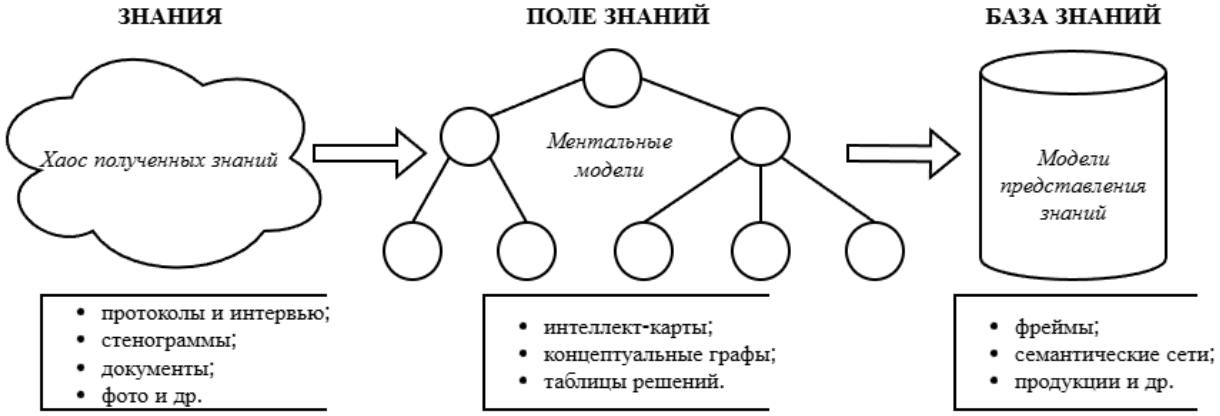

Рисунок 1 – Основные фазы обработки знаний

1.2 Структурирование знаний

Современные подходы к структурированию знаний держатся на трех философских китах: структурализм, семиотика, синергетика.

Результатом процесса структурирования знаний предложено считать **поле знаний**. Поле знаний определено как условное описание основных понятий предметной области и их взаимосвязей в виде графа, диаграммы, таблицы, формул или текста, полученное после завершения процесса извлечения знаний на стадии структурирования [4].

Для структурирования знаний обычно используется следующий алгоритм проведения концептуального анализа знаний:

1) определение входной и выходной информации;
2) составление словаря терминов;
3) выявление объектов, понятий и их атрибутов;
4) выявление связей между понятиями;
5) выделение метапонятий и детализация понятий;
6) построение пирамиды знаний;
7) определение отношений между понятиями;
8) определение стратегии принятия решения.

Структурирование информации позволяет упростить поиск информации, такой как: различные документы, задачи, идеи. Кроме того, структурирование информации позволяет облегчить восприятие и запоминание информации.

1.2.1 Принципы структурирования данных

Структурирование материала подразумевает его упрощение. Нужно разбить сложные логические связи на простые элементы. Рассмотрим следующие принципы структурирования информации.

Выделить несколько групп. Прежде чем составлять структуру данных, важно сформировать представление о том, что необходимо получить в результате, какие данные представляют ценность. Например, у вас стоит задача выполнить анализ конкурентов. Какую информацию о конкурентах вам важно получить? Как минимум это может быть стоимость продукта и его характеристики. Отталкиваясь от этой информации, выделите ключевые группы данных – стоимость и характеристики. При необходимости группы можно дополнительно разбить на подгруппы. Например, если характеристик товара

много, можно их структурировать, создав подгруппы «Материал», «Производитель», «Цвета».

Создать логические связи между группами. Группы должны быть взаимосвязаны и упорядочены относительно друг друга. Например, данные про конкурентов можно распределить в порядке приоритетов – какая информация является самой важной для вас. На данном этапе происходит дополнительная проверка – правильно ли были составлены группы или их нужно изменить. В результате должна получиться упорядоченная схема данных – структура.

Наполнить структуру информацией. Когда структура готова, распределите материал по ней. В зависимости от решаемой задачи одна информация будет для вас важной, а другая нет. Отсеивайте неважные данные. Например, если у вас есть информация об истории развития компании вашего конкурента и она не представляет для вас ценности, ее можно отбросить.

1.2.2 Методы структурирования информации

В зависимости от специфики задачи выбираются разные приемы структурирования информации. Это может быть простая сортировка, распределение по группам или визуальное представление.

Сортировка. Это самый простой способ упорядочить информацию. Его удобно использовать, когда есть огромный объем данных. Например, термины в словаре или имена в телефоне. Люди часто используют сортировку для структурирования данных, даже не замечая этого. Например, многие носят деньги в кошельке, расположив купюры в порядке возрастания номинала (от 5 до 200 рублей).

Отсортировать данные можно по разным критериям:

1) *по алфавиту* (от А до Я). Удобно применять, например, для сортировки списка студентов;

2) *по номерам* (по возрастанию или по убыванию). Так, начальник автопарка может вести список водителей, сортируя его по количеству допущенных нарушений за год;

3) *в хронологическом порядке* (по дате и времени). К примеру, на главной странице блога все статьи отсортированы по дате публикации – от новых к старым.

Классификация. ***Классификация*** – это группировка данных по определенному признаку. Например, документы можно структурировать по

назначению (отчеты, договоры, счета и т. п.) или по дате (январь, февраль и т. п.), рабочие задачи – по проектам, по приоритетности, по исполнителям или по срокам.

Количественная классификация:

1 *По расположению.* Группировать данные по расположению удобно, когда информация прибывает из различных источников или мест действия. Например, если у компании есть филиалы в разных городах, можно создать группы сотрудников по городам.

2 *По количеству.* Если выделить диапазоны значений и придумать для них названия, получим группы. К примеру, все фильмы можно сгруппировать в группы по рейтингу «Высокий», «Средний» и «Низкий». Каждой категории будет соответствовать определенный диапазон оценок.

3 *По времени.* Это лучшая форма классификации для событий, которые происходят в различные временные интервалы. К примеру, вы рассказываете про этапы становления компании и делите всю информацию по годам или вы располагаете документы в папки по месяцам. Иногда такую классификацию удобно визуально отобразить на временном отрезке, чтобы упростить понимание и запоминание ключевых событий.

4 *По категориям.* Данная группировка позволяет объединить данные по общему признаку (цвет, форма, вкус, материал). Такой тип классификации часто используют для товаров и отраслей промышленности. В справочниках легко найти магазины и услуги по категориям.

5 *По иерархии.* При комбинировании разных методов структурирования информации получаем иерархию. Она позволяет построить многоуровневую структуру данных. Например, при структурировании контактов в телефонной книге: рабочие (клиенты, команда, проекты) и личные контакты (знакомые, семья).

Примечание – Все группы должны быть однородными и соответствовать одному критерию. Неправильно будет разделить документы на три группы по разным критериям: «Счета», «Отчеты» и «Новые» (по категориям и по времени). Второй критерий можно использовать только на другом уровне классификации.

Визуализация. Любой материал можно структурировать с помощью визуальных элементов – представить данные в виде графиков, диаграмм,

структурных схем, таблиц и ментальных карт. Остановимся подробнее на последних.

Ментальная или интеллект-карта (Mindmap) – это способ представления информации с помощью блок-схемы. Идея состоит в том, чтобы изобразить центральный объект, от которого расходятся связи-ассоциации, соединяющие его с другими объектами (например, с записями и изображениями). У такой карты может быть бесконечное количество элементов.

Ментальные карты часто используют для многоуровневого структурирования данных, правильной постановки целей и ведения проектов. Для того чтобы нарисовать такую карту, удобно воспользоваться специальным сервисом, например MindMeister [5] или Miro [6]. Некоторые предпочитают рисовать на бумаге от руки.

1.3 Модели представления знания

В настоящее время разработано множество моделей представления знаний. Каждая из них обладает своими преимуществами и недостатками, и поэтому для каждой конкретной задачи необходимо выбирать именно свою модель. От этого будет зависеть не столько эффективность выполнения поставленной задачи, сколько возможность ее решения в принципе. Отметим, что модели представления знаний относятся к прагматическому направлению исследований в области искусственного интеллекта. Это направление основано на предположении о том, что мыслительная деятельность человека – «черный ящик». При таком подходе не ставится вопрос об адекватности используемых в компьютере моделей представления знаний тем моделям, которыми пользуется в аналогичных ситуациях человек, а рассматривается лишь конечный результат решения конкретных задач. Имея обобщенное название, они различаются по идеям, лежащим в их основе, с точки зрения математической обоснованности (рисунок 2).

Рисунок 2 – Модели представления знаний

Эмпирический подход основан на изучении принципов организации человеческой памяти и моделировании механизмов решения задач человеком.

Другой подход – **теоретический**, можно также назвать теоретически обоснованным. Он гарантирует правильность решений. В рамках этого подхода до настоящего времени удавалось решать только сравнительно простые задачи из узкой предметной области.

Кроме моделей, разработанных в рамках каждого из этих подходов, отдельно выделяют (относя к обоим подходам сразу) бионическое направление, представленное **генетическими алгоритмами** и **нейронными сетями**.

Стоит отметить, что в теоретических моделях знания строго формализованы, а в эмпирических призваны дать некоторую свободу.

Рассмотрим более подробно эмпирические модели.

Продукционные модели. ***Продукционная система*** (production system) – это модель представления, основанная на *продукционных правилах* (production rule), позволяющая описывать знания о решении задач в виде правил «ЕСЛИ *условие*, ТО *действие*» (IF …, THEN …). Понятие «продукционные системы» является частным случаем *систем, основанных на знаниях* (knowledge based systems). Впервые идея представления знаний в виде продукций появилась в работе Эмиля Поста (Emil Leon Post) в 1943 г. [4].

В продукционной системе знания представляются в виде продукционной модели, в которой под условием (*антецедентом*) понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний. А под действием (*консеквентом*) – действия, выполняемые при успешном исходе поиска (они могут быть промежуточными, т. е. выступающими далее как условия, и терминальными, или целевыми, завершающими работу системы).

Правила обеспечивают формальный способ представления рекомендаций, указаний или стратегий. Они идеально подходят в тех случаях, когда знания предметной области возникают из эмпирических ассоциаций, накопленных за годы работы по решению задач в некоторой предметной области [4].

Чаще всего вывод на базе знаний, в основе которой продукционная модель, бывает:

− *прямой* (от данных к поиску цели) – это поиск действия по заданному условию;

− *обратный* (от цели (для ее подтверждения) к данным) – поиск возможных условий, которые могли бы привести к указанному действию.

Продукционная модель чаще всего применяется в промышленных экспертных системах.

Семантические сети. Термин ***семантическая*** означает «смысловая», а сама ***семантика*** – это наука, устанавливающая отношения между символами и объектами, которые они обозначают, т. е. наука, определяющая смысл знаков.

Семантическая сеть – это ориентированный граф, *вершины* которого – понятия, а *дуги* – отношения между ними.

Семантическая модель позволяет снизить объем хранимых данных, обеспечивает реализацию ассоциативных связей. Проблема гибкости модели и существования бесконечного множества возможных связей решается добавлением новых типов отношений. В проектах, посвященных моделированию человеческой памяти, было предложено при использовании расширяемой подобным образом семантической сети также строить модель, хранящую все типы соединений и отношения подобия и взаимозаменяемости между ними.

В качестве понятий обычно выступают абстрактные или конкретные *объекты*, а *отношения* – это связи типа: «это» («AKO – a-kind-of», «is»), «имеет частью» («has part»), «принадлежит», «любит».

Характерной особенностью семантических сетей является обязательное наличие трех типов отношений:

− **класс** – элемент класса (цветок – роза);

− **свойство** – значение (цвет – желтый);

− **пример элемента класса** (роза – чайная).

Наиболее часто в семантических сетях используются следующие отношения:

− связь типа «часть – целое», «класс – подкласс», «элемент – множество» и т. п.;

− функциональные связи (определяемые обычно глаголами «производит», «влияет» и т. д.);

− количественные (больше, меньше, равно и т. д.);

− пространственные (далеко от, близко от, за, под, над и т. д.);

− временные (раньше, позже, в течение и т. д.);

− атрибутивные связи (иметь свойство, иметь значение);

− логические связи (И, ИЛИ, НЕ);

− лингвистические связи.

Узлы в семантической сети обычно соответствуют объектам, концепциям, событиям или понятиям. Любой фрагмент сети, например одна вершина, две вершины и соединяющие их дуги, называют ***подсетью***. Логический вывод (поиск решения) на семантической сети заключается в том, чтобы найти или сконструировать подсеть, удовлетворяющую некоторым условиям.

Отношения, представляемые дугами, в семантической сети могут быть различными (таблица 1). Типы отношений выбираются в зависимости от вида семантической сети (таблица 2) и решаемой задачи.

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, отражающей поставленный запрос к базе.

Таблица 1 – Основные виды отношений в семантических сетях

Тип	**Описание**
1	**2**
Являться наследником (a-kind-of)	Задает иерархические связи между классами
Являться экземпляром (IS-A)	Определяет значение, описывает конкретный объект, понятие

Продолжение таблицы 1

1	2
Это (ARE, есть)	Может использоваться вместо связи a-kind-of в отношениях, подразумевающих равенство или эквивалентность
Являться частью (HAS-PART)	Определяет структурные связи, описывает части или целые объекты
Функциональные	Определяются обычно глаголами, отражают различные отношения (учить, владеть и т. д.)
Количественные	Отображают количественные соотношения между вершинами (больше, меньше и т. д.)
Пространственные	Отображают пространственные отношения между вершинами (близко, далеко и т. д.)
Временные	Описывают временные связи между вершинами (скоро, долго, сейчас и т. д.)
Атрибутивные	Описывают свойства объектов, понятий
Логические	Описывают логические связи между вершинами (И, ИЛИ, НЕ)

Таблица 2 – Типы семантических сетей

Тип	Описание
1	**2**
По типу знания	
Экстенсиональные	Описывает конкретные отношения данной ситуации
Интенсиональные	Описывают имена классов объектов, а не индивидуальные имена объектов; связи отражают те отношения, которые всегда присущи объектам данного класса
По типу ограничений на дуги и вершины	
Простые	Вершины сети не обладают внутренней структурой
Иерархические	Вершины обладают внутренней структурой; в иерархической сети есть возможность разделять сеть на подсети и устанавливать отношения не только между вершинами, но и между подсетями (различные подсети, существующие в сети, могут быть упорядочены в виде дерева подсетей, вершины которого – подсети, а дуги – отношения видимости)
Динамические (сценарии)	Сети с событиями
По количеству типов отношений	
Однородные	Обладают только одним типом отношений
Неоднородные	Количество типов отношений больше двух

Продолжение таблицы 2

1	2
По арности отношений	
Бинарные	Все отношения в графе связывают ровно два понятия
N-арные	В сети есть отношения, связывающие более двух объектов

Фреймы. Фреймом называется формализованная модель для отображения образа.

Термин ***фрейм*** (от англ. Frame – «каркас», «рамка») был предложен Марвином Минским в 1979 г. для обозначения структуры знаний для восприятия пространственных сцен и является развитием семантических сетей. Другими словами, ***фрейм*** – это абстрактный образ для представления некоего стереотипа восприятия [3, 4].

В психологии и философии известно понятие абстрактного образа. Например, произнесение вслух слова «комната» порождает у слушающих образ комнаты: «жилое помещение с четырьмя стенами, полом, потолком, окнами и дверью, площадью 6–20 м^2»

Из этого описания ничего нельзя убрать (например, убрав окна, мы получим уже чулан, а не комнату), но в нем есть «дырки», или «слоты» – это незаполненные значения некоторых атрибутов – например: количество окон, цвет стен, высота потолка, покрытие пола и др.

В теории фреймов такой образ комнаты называется фреймом комнаты.

Структура фрейма состоит из следующей информации:

– *неинициализированные значения*. Слоты могут оставаться незаполненными, пока они не понадобятся для решения какой-либо задачи. Например, направления следования поездов являются обязательным атрибутом каждого вокзала, но эти направления не могут быть определены, если неизвестно, о каком конкретно вокзале идет речь;

– *метаинформация* – информация о допустимых значениях для заполнения слотов.

В качестве идентификатора фрейму присваивается имя. Это имя должно быть единственным во всей фреймовой системе. А также каждый фрейм имеет список слотов и их значений [3, 4].

Значениями могут быть данные любого типа, а также название другого фрейма. Таким образом, фреймы образуют сеть. Кроме того, существует связь

между фреймами типа ***АКО*** **(**a-kind-of**)**, которая указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются список и значения слотов. При этом возможно ***множественное наследование*** – перенос свойств от нескольких прототипов.

Фрейм отражает основные свойства объекта или явления. Информация во фреймах записывается в виде списка свойств, называемых во фрейме слотами (англ. slot – «паз», «щель»). Таким образом, слот является основной структурной единицей фрейма. ***Слоты*** – это некоторые незаполненные подструктуры фрейма, заполнение которых приводит к тому, что данный фрейм ставится в соответствие некоторой ситуации, явлению или объекту.

Слот представляет собой пару: имя слота и его значение. В качестве значения слота могут выступать константы (факты), выражения с переменными, ссылки на другие слоты и т. п.

Слот может иметь структуру, элементы которой сами являются слотами. Фрейм состоит из конечного значения слотов. Слотам фреймов могут быть приписаны по умолчанию некоторые стандартные значения. Значения, присвоенные по умолчанию, могут быть заменены значениями, подходящими для обрабатываемой ситуации.

Любой фрейм может быть представлен следующим образом:

(ИМЯ ФРЕЙМА:
(имя 1-го слота: значение 1-го слота),
(имя 2-го слота: значение 2-го слота),
……..
(имя N-го слота: значение N-го слота)).

Табличное представление слота приведено ниже (таблица 3).

Таблица 3 – Структура фрейма

ИМЯ ФРЕЙМА			
Имя слота	*Значение слота*	*Способ получения значения*	*Демон*

В таблице 3 дополнительные столбцы предназначены для описания способа получения слотом его значения и возможного присоединения к тому или иному слоту специальных процедур, что допускается в теории фреймов. В качестве значения слота может выступать имя другого фрейма; так образуют сети фреймов.

Существует несколько способов получения значения (таблица 4). Они определяют, как именно устанавливается значение конкретного слота. Выбор способа зависит от свойств самих данных.

Таблица 4 – Способы получения значений слотов

Способ	**Описание**
По умолчанию от прототипа (родителя)	Слоту присваивается значение, определенное по умолчанию во фрейме-прототипе, а также некоторые стандартные значения
Через наследование	Значение задано в специальном слоте родительского фрейма, соединенного с текущим связью АКО
По формуле	Слоту назначается формула, результат вычисления которой является значением слота
Через присоединенную процедуру	Слоту назначается процедура, позволяющая получить значение слота алгоритмически
Из внешних источников данных	При использовании модели в интеллектуальных системах данные, являющиеся значениями слотов, могут поступать из баз данных, от системы датчиков, от пользователя

Существует несколько видов фреймов, которые позволяют описать предметную область и решаемую задачу.

Различают *фреймы-образцы*, или прототипы, хранящиеся в базе знаний, и *фреймы-экземпляры*, которые создаются для отображения реальных фактических ситуаций на основе поступающих данных. ***Фрейм-прототип*** – это интенсиональное описание некоторого множества фреймов-примеров.

В таблице 5 представлены наиболее распространенные типы фреймов, указаны типы знаний, которые они отображают, а также примеры фреймов данного типа из различных предметных областей.

Таблица 5 – Типы фреймов

Тип фрейма	**Тип знания**	**Описание**	**Пример**
1	**2**	**3**	**4**
По познавательному назначению			
Фреймы-прототипы (шаблоны, образцы)	Интенсиональные	Отражают знания об абстрактных стереотипных понятиях, которые являются классами каких-либо конкретных объектов	Человек, автомобиль

Продолжение таблицы 5

1	2	3	4
Фреймы-экземпляры (примеры)	Экстенсиональные	Отражают знания о конкретных фактах из предметной области	Иванов И. И., ВАЗ-2110
По функциональному назначению			
Фреймы-структуры (объекты)	Декларативные	Отображают абстрактные и конкретные предметы и понятия предметной области (содержат набор характеристик, описывающий объект или понятие)	Заем, залог, вексель, человек, лекция
Фреймы-операции	Процедурные	Отображают различные процессы преобразования или использования объектов предметной области (содержат набор характеристик процесса)	Процессы получения займа, синтеза устройств
Фреймы-ситуации	Прагматические	Отображают типичные ситуации, в которых могут находиться фреймы-объекты и фреймы-роли (содержат набор характеристик, идентифицирующих ситуацию)	Авария, тревога, рабочий режим устройства
Фреймы-сценарии	Технологические	Отображают развитие ситуации, типовую структуру для некоторого действия, понятия, события, отображают динамику (содержат набор характеристик, позволяющих обеспечить развитие системы по данному сценарию)	Банкротство, празднование именин, сдача экзамена
Фреймы-роли	Функциональные	Отображают типичную роль, выполняемую фреймом-объектом в определенной ситуации (содержат набор характеристик роли)	Менеджер, кассир, клиент, студент, преподаватель

Важнейшим свойством теории фреймов является заимствованное из теории семантических сетей наследование свойств. И во фреймах, и в семантических сетях наследование происходит по *АКО-связям*. Слот АКО

указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, т. е. переносятся, значения аналогичных слотов.

Значение слота. Оно должно соответствовать указанному типу данных и условию наследования. Значением слота может быть практически что угодно: числа, формулы, тексты на естественном языке или программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов. В качестве значения слота может выступать набор слотов более низкого уровня, что позволяет реализовывать во фреймовых представлениях «принцип матрешки». Связи между фреймами задаются значениями специального слота с именем «Связь».

В общем случае структура данных фрейма может содержать более широкий набор информации, в который входят следующие атрибуты.

Имя фрейма. Оно служит для идентификации фрейма в системе и должно быть уникальным. Фрейм представляет собой совокупность слотов, число которых может быть произвольным. Число слотов в каждом фрейме устанавливается проектировщиком системы, при этом часть слотов определяется самой системой для выполнения специфических функций (системные слоты), примерами которых являются: слот-указатель родителя данного фрейма (IS-A), слот-указатель дочерних фреймов, слот для ввода имени пользователя, слот для ввода даты определения фрейма, слот для ввода даты изменения фрейма и т. д.

Имя слота. Оно должно быть уникальным в пределах фрейма. Обычно имя слота представляет собой идентификатор, который наделен определенной семантикой. В качестве имени слота может выступать произвольный текст. Например, <Имя слота> = Главный герой романа Ф. М. Достоевского «Идиот», <Значение слота> = Князь Мышкин. Имена системных слотов обычно зарезервированы, в различных системах они могут иметь различные значения. Примеры имен системных слотов: IS-A, HAS-PART, RELATIONS и т. д. Системные слоты служат для редактирования базы знаний и управления выводом во фреймовой системе.

Указатели наследования. Они показывают, какую информацию об атрибутах слотов из фрейма верхнего уровня наследуют слоты с аналогичными именами в данном фрейме. Указатели наследования характерны для фреймовых систем иерархического типа, основанных на отношениях типа «абстрактное –

конкретное». В конкретных системах указатели наследования могут быть организованы различными способами и иметь разные обозначения:

− *U (Unique)* – значение слота не наследуется;

− *S (Same)* – значение слота наследуется;

− *R (Range)* – значения слота должны находиться в пределах интервала значений, указанных в одноименном слоте родительского фрейма;

− *O (Override)* – при отсутствии значения в текущем слоте оно наследуется из фрейма верхнего уровня, однако в случае определения значения текущего слота оно может быть уникальным. Этот тип указателя выполняет одновременно функции указателей U и S.

Указатель типа данных. Он показывает тип значения слота. Наиболее употребляемые типы:

− *frame* – указатель на фрейм;

− *real* – вещественное число;

− *integer* – целое число;

− *boolean* – логический тип;

− *text* – фрагмент текста;

− *list* – список;

− *table* – таблица;

− *expression* – выражение;

− *lisp* – связанная процедура и т. д.

Демоны (таблица 6). ***Демоном*** называется процедура, автоматически запускаемая при выполнении некоторого условия. Демонов может быть несколько. Наиболее близок механизм присоединенных процедур к триггерам в реляционных базах данных. Демоны автоматически запускаются при обращении к соответствующему слоту. Типы демонов связаны с условием запуска процедуры. Демон с условием IF-NEEDED запускается, если в момент обращения к слоту его значение не было установлено. Демон типа IF-ADDED запускается при попытке изменения значения слота. Демон IF-REMOVED запускается при попытке удаления значения слота. Возможны также другие типы демонов. Демон является разновидностью связанной процедуры.

Таблица 6 – Наиболее распространенные демоны

Демон	Событие	Описание
IF-REMOVED	Если удалено	Выполняется, когда информация удаляется из слота
IF-ADDED	Если добавлено	Выполняется, когда новая информация записывается в слот
IF-NEEDED	По требованию	Выполняется, когда запрашивается информация из пустого слота
IF-DEFAULT	По умолчанию	Выполняется, когда устанавливается значение по умолчанию

Присоединенная процедура. В качестве значения слота может использоваться процедура, называемая служебной в языке Лисп (Lisp) или методом в языках объектно-ориентированного программирования. Присоединенная процедура запускается по сообщению, переданному из другого фрейма. Демоны и присоединенные процедуры являются процедурными знаниями, объединенными вместе с декларативными в единую систему. Эти процедурные знания являются средствами управления выводом во фреймовых системах, причем с их помощью можно реализовать любой механизм вывода. Представление таких знаний и заполнение ими интеллектуальных систем – весьма нелегкое дело, которое требует дополнительных затрат сил и времени разработчиков. Поэтому проектирование фреймовых систем выполняется, как правило, специалистами, имеющими высокий уровень квалификации в области искусственного интеллекта.

Основным преимуществом фреймов как модели представления знаний является то, что она отражает концептуальную основу организации памяти человека, а также ее гибкость и наглядность. Наиболее ярко достоинства фреймовых систем представления знаний проявляются в том случае, если родовидовые связи изменяются нечасто и предметная область насчитывает немного исключений. Во фреймовых системах данные о родовидовых связях хранятся явно, как и знания других типов. Значения слотов представляются в системе в единственном экземпляре, поскольку включаются только в один фрейм, описывающий понятия, которые содержит слот с данным именем. Такое свойство систем фреймов обеспечивает экономное размещение базы знаний в памяти компьютера. Еще одно достоинство фреймов состоит в том, что значение любого слота может быть вычислено с помощью соответствующих процедур или

найдено эвристическими методами. То есть фреймы позволяют манипулировать как декларативными, так и процедурными знаниями.

К недостаткам фреймовых систем относят их относительно высокую сложность, что проявляется в снижении скорости работы механизма вывода и увеличении трудоемкости внесения изменений в родовую иерархию. Поэтому при разработке фреймовых систем уделяют внимание наглядным способам отображения и эффективным средствам редактирования фреймовых структур.

Ленемы. ***Ленемы*** представляют собой смешанный тип модели, являющийся как бы «развитием» других моделей (фреймы, семантические сети и т. д.). Ленема предназначена для структурного комплексного описания понятий предметной области. По изобразительным возможностям ленемы более совершенны, чем такие традиционные модели представления знаний, как семантическая сеть, фрейм, система продукций. Однако, для некоторых понятий модель представления знаний на основе ленем может быть неудобной и даже неприемлемой.

Например, это такие понятия, в описании которых очень большую роль играет внутренняя динамика. Модель, созданная на базе ленем, позволяет объединить на пользовательском уровне три существующие в настоящее время парадигмы представления знаний:

− логическую (продукционная и логическая модели);

− структурную (семантические сети и фреймы);

− процедурную.

Для некоторых ситуаций это очень удобно, так как при реализации сложных моделей, включающих знания различных типов, возникает необходимость совмещения в одном языке представления знаний различных концепций.

Теоретически обоснованный подход гарантирует правильность решений. Данный подход в основном представлен моделями, основанными на формальной логике (исчисление высказываний, исчисление предикатов), формальных грамматиках, комбинаторными моделями, в частности моделями конечных проективных геометрий, теории графов, тензорными и алгебраическими моделями. Рассмотрим модели, относящиеся к теоретическому подходу более подробно.

Логическая модель. Вся информация в логической модели рассматривается как совокупность фактов и связывающих их утверждений, которые представляются как формулы в некоторой логике. Знания при этом представляются набором подобных утверждений, а построение выводов и получение новых знаний сводится к реализации процедуры логического вывода. Этот процесс может быть строго формализован, так как в его основе лежит классический аппарат математической логики.

Формальные грамматики. Формальная грамматика (теория) состоит из алфавита (словаря), множества синтаксических правил, которые позволяют определить истинность или ложность выражений, построенных в данном языке, базовой системы подобных выражений, которые всегда истинны и называются ***аксиомами***, множества правил вывода, позволяющих преобразовать одно выражение в другое.

В основе этой модели лежит исчисление высказываний, которое можно считать классическим примером аксиоматических систем. Эта система хорошо исследована и имеет разработанную модель логического вывода. Эти свойства переносятся и на модель, ее использующую.

Главным недостатком является отсутствие гибкости системы. В случае модификации или расширения модели может потребоваться перестроить всю систему, что для практических систем неприемлемо. Как следствие, формальные грамматики используются в тех предметных областях, которые хорошо локализуются и мало зависят от внешних факторов.

Комбинаторные модели. Комбинаторные модели основаны на рассмотрении дискретных объектов, конечных множеств и заданном на них отношении порядка. Комбинаторика также рассматривает все возможные изменения, перестановки и сочетания в рамках заданных множеств.

Комбинаторные модели используются в задачах топологии (например, поиск пути), прогнозирования поведения автоматов, а также при изучении деревьев решений, частично упорядоченных множеств.

Основная проблема указана еще в определении этой модели: она оперирует только дискретными объектами и конечными множествами, связанными однородными отношениями.

Алгебраические модели. Алгебраическая модель подразумевает представление знаний в виде некоторых алгебраических примитивов, над которыми определено множество действий (некоторые из которых можно задать таблично). Для набора знаний, представленного в таком виде, действуют правила алгебраических множеств, такие как аксиоматизация, определение подсистем и отношений эквивалентности. Также возможно построение цепей множеств (множества, для которых определен порядок отношения «быть подсистемой»).

Нейронные сети, генетические алгоритмы. Данные модели нельзя строго отнести к эмпирическому или теоретическому подходам. Их относят к бионическому направлению. Оно основывается на предположении о том, что если в искусственной системе воспроизвести структуры и процессы человеческого мозга, то и результаты решения задач такой системой будут подобны результатам, получаемым человеком.

Рекомендации по выполнению практических заданий

Выбор предметной области. Предметная область для выполнения практических заданий может быть выбрана на усмотрение обучаемого. Одна предметная область выбирается для выполнения всех четырех практических заданий. Недопустимо повторение одной предметной области в рамках группы.

Предлагаемые предметные области для выполнения практических заданий:

1 «Аэропорт» (диспетчерская).

2 «Железная дорога» (продажа билетов).

3 «Торговый центр» (организация).

4 «Продуктовый магазин» (организация).

5 «Автозаправка» (обслуживание клиентов).

6 «Автопарк» (пассажирские перевозки).

7 «Компьютерные сети» (организация).

8 «Университет» (учебный процесс).

9 «Парикмахерская/барбершоп» (обслуживание клиентов).

10 «Компьютерная безопасность» (средства и способы ее обеспечения).

11 «Компьютерная безопасность» (угрозы).

12 «Интернет-кафе» (организация и обслуживание).

13 «Разработка информационных систем» (ведение информационного проекта).

14 «Туристическое агентство» (работа с клиентами).

15 «Зоопарк» (организация).

16 «Кухня» (приготовление пищи).

17 «Больница» (прием больных).

18 «Кинопрокат» (ассортимент и работа с клиентами).

19 «Прокат автомобилей» (ассортимент и работа с клиентами).

20 «Операционные системы» (функционирование).

21 «Информационные системы» (виды и функционирование).

22 «Предприятие» (структура и функционирование).

23 «Банковское дело» (структура и функционирование).

24 «Животный мир» (структура и взаимодействие).

25 «Религии» (структура и функционирование).

26 «Неисправности ЭВМ» (классификация, пути устранения).

27 «Подбор персонала» (проведение интервью, порядок трудоустройства, отказ, знакомство с организацией).

28 «Печатная продукция» (классификация, применение).

29 «Флористика» (разновидности и флористические техники).

30 «Организация времени» (распределение задач и их классификация).

Оформление результатов. Каждое практическое задание оформляется в виде отчета. Рекомендуется использовать следующую структуру:

− титульный лист с указанием номера практического задания и проектируемой предметной области;

− непосредственная реализация в соответствии с заданием предметной области;

− выводы.

Практическое задание № 1. Анализ предметной области

Цель: научиться структурировать информацию.

Задание. Представить выбранную предметную область в виде интеллектуальной карты (ментальной карты).

Описание процесса решения:

1 Выбрать предметную область для структуризации.

2 Выделить критерии для структуризации.

3 Построить ментальную карту выбранной предметной области с помощью удобного для вас сервиса.

4 Оформить отчет. В отчете необходимо дать краткую характеристику предметной области, перечислить критерии структуризации, представить полученную ментальную карту предметной области.

Пример решения задачи

Задача. Построить в виде ментальной карты фрагмент знаний *предметной области «Ресторан» (посещение ресторана)*.

Решение.

1 Определим любимые места для посещения.

2 Отметим какой был заказ.

3 Запомним как зовут официанта.

4 Зафиксируем адрес.

5 Запишем мнение.

6 И так будем делать с каждым рестораном, который понравился. Результат представлен на рисунке 3.

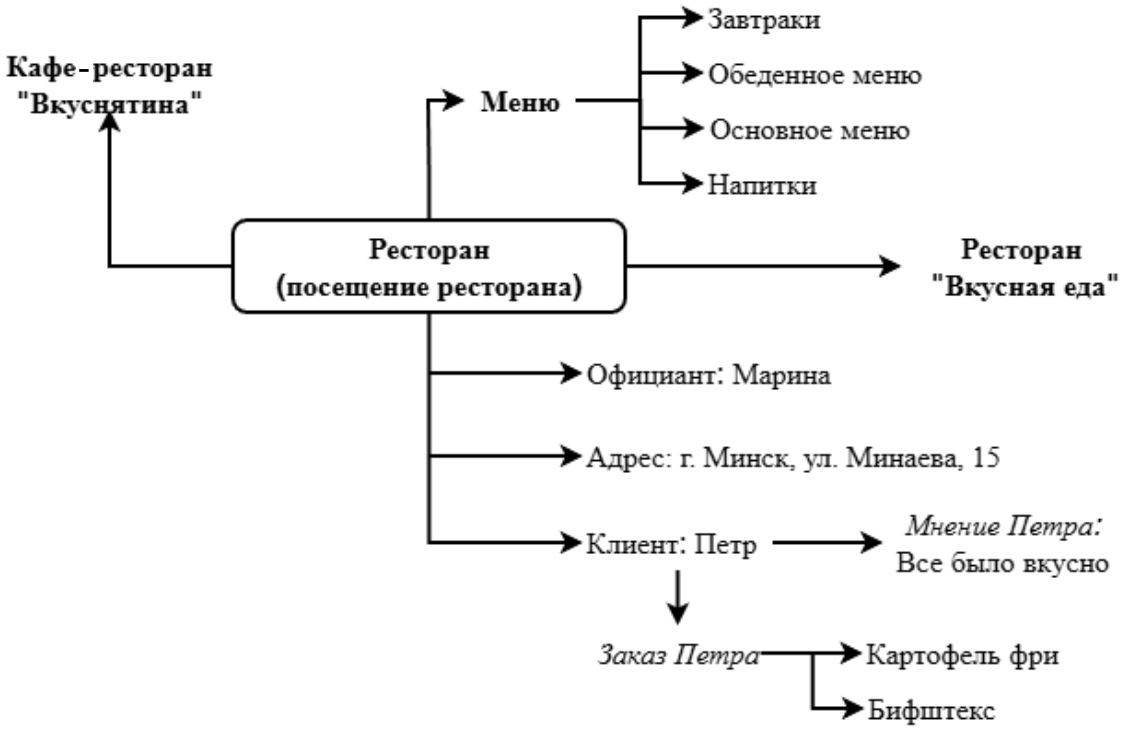

Рисунок 3 – Пример представления предметной области «Ресторан» в виде ментальной карты

Практическое задание № 2. Представление знаний в виде семантической сети

Цель: научиться представлять знания с помощью семантической сети.

Задание. Построить сетевую модель представления знаний выбранной предметной области.

Описание процесса решения. Для построения сетевой модели представления знаний необходимо выполнить следующие шаги.

1 Определить абстрактные объекты и понятия предметной области, необходимые для решения поставленной задачи. Оформить их в виде вершин.

2 Задать свойства для выделенных вершин, оформив их в виде вершин, связанных с исходными вершинами атрибутивными отношениями.

3 Задать связи между этими вершинами, используя функциональные, пространственные, количественные, логические, временные, атрибутивные отношения, а также отношения типа «являться наследником» и «являться частью».

4 Добавить конкретные объекты и понятия, описывающие решаемую задачу. Оформить их в виде вершин, связанных с уже существующими отношениями типа «являться экземпляром», «есть».

5 Проверить правильность установленных отношений (вершины и само отношение при правильном построении образуют предложение, например «Двигатель является частью автомобиля»).

Пример решения задачи

Задача. Построить семантическую сеть фрагмента знаний *предметной области «Ресторан» (посещение ресторана)*.

Решение.

1 Ключевые понятия данной предметной области – ресторан, тот, кто посещает ресторан (клиент), и те, кто его обслуживают (повара, метрдотели, официанты, для простоты ограничимся только официантами). У обслуживающего персонала и клиентов есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Продукцией ресторана являются блюда, которые заказывают клиенты. Исходя из этого, вершины графа будут следующими: «Ресторан», «Человек», «Официант», «Клиент», «Заказ» и «Блюдо».

2 У этих объектов есть определенные свойства и атрибуты. Например, рестораны располагаются по определенным адресам, каждое блюдо из меню имеет свою цену. Поэтому добавим вершины «Адрес» и «Цена».

3 Определим для имеющихся вершин отношения и их типы, используя таблицы 1 и 2 (см. с. 16–18).

4 Добавим знание о конкретных фактах решаемой задачи. Пусть имеется два ресторана: «Вкуснятина» и «Вкусная еда», в первом работает официантка Марина, а во втором – официант Сергей. Петр решил пойти в ресторан «Вкусная еда» и сделал заказ официанту на два блюда: картофель фри за 30 рублей, бифштекс за 130 рублей. Также известны адреса этих ресторанов и их специфика.

Исходя из этого, добавим соответствующие вершины в граф и соединим их функциональными отношениями и отношениями типа «например или являться экземпляром». Полученный в результате граф изображен на рисунке 4.

5 Осуществим проверку установленных связей. Например, возьмем вершину «Блюдо» и пройдем по установленным связям. Получаем следующую информацию: блюдо является частью заказа, примерами блюд могут служить картофель фри и бифштекс.

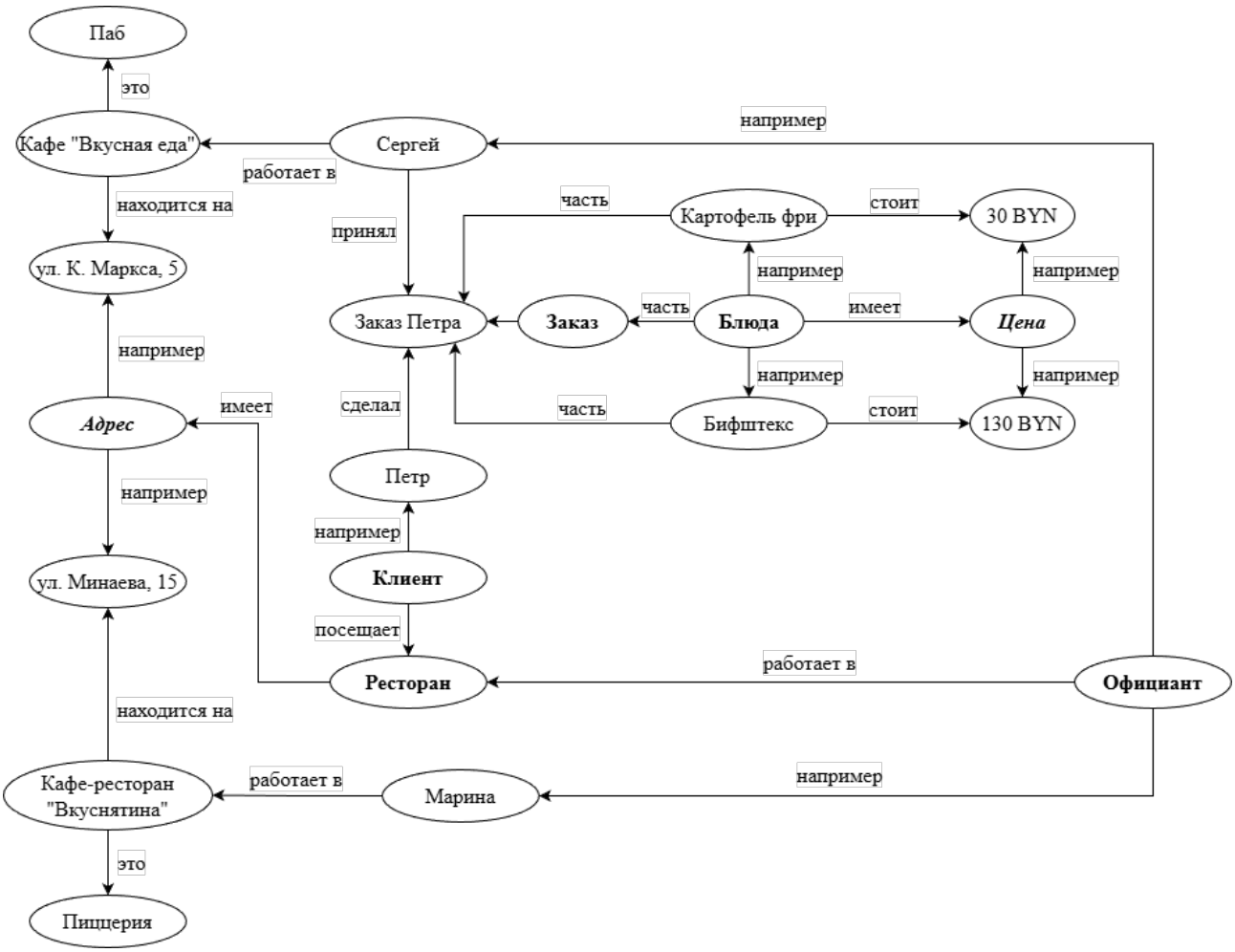

Рисунок 4 – Семантическая сеть предметной области «Ресторан»

Практическое задание № 3. Представление знаний в виде фреймовой модели

Цель: научиться представлять знания с помощью фреймов.

Задание. Представить выбранную предметную область с помощью фреймовой модели представления знаний.

Описание процесса решения. Для построения фреймовой модели необходимо выполнить следующие шаги.

1 Определить абстрактные объекты и понятия предметной области, необходимые для решения поставленной задачи. Оформить их в виде фреймов-прототипов (фреймов-объектов, фреймов-ролей).

2 Задать конкретные объекты предметной области. Оформить их в виде фреймов-экземпляров (фреймов-объектов, фреймов-ролей).

3 Определить набор возможных ситуаций. Оформить их в виде фреймов-ситуаций (прототипы). Если существуют прецеденты по ситуациям в предметной области, добавить фреймы-экземпляры (фреймы-ситуации).

4 Описать динамику развития ситуаций (переход от одних к другим) через набор сцен. Оформить их в виде фреймов-сценариев.

5 Добавить фреймы-объекты сценариев и сцен, которые отражают данные конкретной задачи.

6 Графически изобразить взаимодействие между собой описанных фреймов.

Пример решения задачи

Задача. Построить фреймовую модель представления знаний *в предметной области «Ресторан» (посещение ресторана)*.

Решение.

1 Ключевые понятия данной предметной области: ресторан; тот, кто посещает ресторан – клиент; те, кто обслуживают ресторан (повара, метрдотели, официанты, для простоты ограничимся только официантами) – обслуживающий персонал.

У обслуживающего персонала и клиентов есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Тогда фреймы «Человек» (таблица 7) и «Ресторан» (таблица 8) являются прототипами-образцами, а фреймы «Официант» (таблица 9) и «Клиент» (таблица 10) – прототипами-ролями. Также нужно определить основные слоты фреймов – характеристики, имеющие значения для решаемой задачи.

Таблица 7 – Фрейм-прототип-образец «Человек»

ЧЕЛОВЕК			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Пол	Мужской, женский	Из внешних источников	–
Возраст	От 0 до 120	Из внешних источников	–

Таблица 8 – Фрейм-прототип-образец «Ресторан»

РЕСТОРАН			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
1	**2**	**3**	**4**
Название	[текст]	Из внешних источников	–

Продолжение таблицы 8

1	2	3	4
Адрес	[текст]	Из внешних источников	–
Часы работы	[день недели] [время]	Из внешних источников	–
Специализация	[текст]	Из внешних источников	–
Класс	Средний, Высший	–	–

Таблица 9 – Фрейм-прототип-роль «Официант»

ОФИЦИАНТ (АКО «ЧЕЛОВЕК»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Возраст	От 18 до 55 лет	Из внешних источников	–
Пол	Мужской, женский	Из внешних источников	–
Стаж работы	От 1 до 20	Из внешних источников	–
Зарплата	От 100 до 2000	Из внешних источников	–
График работы	[день недели] [время]	Из внешних источников	–
Место работы	Фрейм-объект	Из внешних источников	–

Таблица 10 – Фрейм-прототип-роль «Клиент»

КЛИЕНТ (АКО «ЧЕЛОВЕК»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Вид оплаты	Наличные, банковская карта	По умолчанию (наличные)	–
Статус	Обычный, VIP	По умолчанию (обычный)	–
Форма заказа	Заказ есть, заказа нет	По умолчанию (заказа нет)	–
Чаевые	От 1 до 10	Из внешних источников	–

2 *Фреймы-наследники* содержат все слоты своих родителей, они явно прописываются только в случае изменения какого-либо параметра.

3 *Фреймы-образцы* описывают конкретную ситуацию: какие рестораны имеются в городе, как именно организовывается посещение, кто является посетителем, кто работает в выбранном ресторане и т. д. Поэтому определим фреймы-образцы, являющиеся наследниками фреймов-прототипов (таблицы 11–15).

Таблица 11 – Фрейм-образец «кафе-ресторан “Вкуснятина”»

КАФЕ-РЕСТОРАН «ВКУСНЯТИНА» (АКО «РЕСТОРАН»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Название	Вкуснятина	Из внешних источников	–
Адрес	г. Минск, ул. Минаева, 15	Из внешних источников	–
Часы работы	09:00–00:00	Из внешних источников	–
Специализация	Пиццерия	Из внешних источников	–
Класс	Средний, высший	Из внешних источников	–

Таблица 12 – Фрейм-образец «кафе “Вкусная еда”»

КАФЕ «Вкусная еда» (АКО «РЕСТОРАН»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Название	Вкусная еда	Из внешних источников	–
Адрес	г. Минск, ул. Карла Маркса, 5	Из внешних источников	–
Часы работы	09:00–00:00	Из внешних источников	–
Специализация	Паб	Из внешних источников	–
Класс	Средний	Из внешних источников	–

Таблица 13 – Фрейм-образец «официант Сергей»

СЕРГЕЙ (АКО «ОФИЦИАНТ»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Возраст	27	Из внешних источников	–
Пол	Мужской	Из внешних источников	–
Стаж работы	5	Из внешних источников	–
Зарплата	700	Из внешних источников	–
График работы	Через день с 18:00 до 00:00	Из внешних источников	–
Место работы	Кафе «Вкусная еда»	Из внешних источников	–

Таблица 14 – Фрейм-образец «официант Марина»

МАРИНА (АКО «ОФИЦИАНТ»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
1	**2**	**3**	**4**
Возраст	24	Из внешних источников	–
Пол	Женский	Из внешних источников	–
Стаж работы	2	Из внешних источников	–

Продолжение таблицы 14

1	2	3	4
Зарплата	820	Из внешних источников	–
График работы	Через день с 09:00 до 14:00	Из внешних источников	–
Место работы	Кафе-ресторан «Вкуснятина»	Из внешних источников	–

Таблица 15 – Фрейм-образец «клиент Петр»

КЛИЕНТ (АКО «ЧЕЛОВЕК»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Пол	Мужской	Из внешних источников	–
Возраст	19	Из внешних источников	–
Вид оплаты	Наличные	По умолчанию (наличные)	–
Статус	Обычный	По умолчанию (обычный)	–
Форма заказа	Заказа нет	По умолчанию (заказа нет)	–
Чаевые	7 % от суммы заказа	Из внешних источников	–

4 *Фреймы-ситуации* описывают возможные ситуации. В ресторане клиент попадает в несколько типичных ситуаций: заказ и оплата. Возможны и другие, не типичные ситуации: клиент подавился, у клиента нет наличных для оплаты счета и т. д. В таблицах 16 и 17 рассмотрены типичные ситуации (их может быть больше) в рамках исследуемой задачи.

Таблица 16 – Фрейм-ситуация «Заказ»

ЗАКАЗ			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Перечень блюд	[текст]	Из внешних источников	–
Перечень цен	От 0 до 100000	Присоединенная процедура	IF-ADDED (изменяет слот «Перечень цен»)
Сумма заказа	От 0 до 100000	Присоединенная процедура	IF-ADDED (изменяет слот «Сумма заказа»)
Принял заказ	Фрейм-образец	Из внешних источников	–
Сделал заказ	Фрейм-образец	Из внешних источников	–

Таблица 17 – Фрейм-ситуация «Оплата»

ОПЛАТА			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Вид платежа	Наличные, карта	Из внешних источников	IF-ADDED (изменяет слот «Чаевые»)
Чаевые	От 0 до 100000	Присоединенная процедура	–
Оплатил	Фрейм-образец	Присоединенная процедура	–
Заказ	Фрейм-образец	Из внешних источников	IF-ADDED (изменяет слот «Оплатил»)

5 *Ситуации* возникают после наступления каких-то событий, выполнения условий и могут следовать одна за другой. Динамику предметной области можно отобразить во *фреймах-сценариях*. Их может быть множество, опишем наиболее общий и типичный сценарий посещения ресторана (таблица 18).

Таблица 18 – Фрейм-сценарий «Посещение ресторана»

ПОСЕЩЕНИЕ РЕСТОРАНА			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Посетитель	Фрейм-объект	Из внешних источников	–
Ресторан	Фрейм-объект	Из внешних источников	IF-ADDED (изменяет слот «Официант»), IF-REMOVED (изменяет слот «Официант»)
Официант	Фрейм-объект	Присоединенная процедура (определяет по выбранному ресторану)	–
Сцена 1	Вход, выбор	Из внешних источников	–
Сцена 2	Заказ	Из внешних источников	–
Сцена 3	Еда	Из внешних источников	–
Сцена 4	Оплата	Из внешних источников	–
Сцена 5	Выход	Из внешних источников	–

6 Пусть в рамках нашей задачи Петр посетил ресторан «Вкусная еда». Тогда фреймы будут заполнены, как показано в таблицах 19–21.

Таблица 19 – Фрейм-сценарий «посещение кафе “Вкусная еда”»

ПОСЕЩЕНИЕ КАФЕ «ВКУСНАЯ ЕДА» (АКО «ПОСЕЩЕНИЕ РЕСТОРАНА»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Посетитель	Петр	Из внешних источников	–
Ресторан	Кафе «Вкусная еда»	Из внешних источников	IF-ADDED (изменяет слот «Официант»), IF-REMOVED (изменяет слот «Официант»)
Официант	Сергей	Присоединенная процедура (определяет по выбранному ресторану)	–
Сцена 1	Вход, выбор	Из внешних источников	–
Сцена 2	Заказ	Из внешних источников	–
Сцена 3	Еда	Из внешних источников	–
Сцена 4	Оплата	Из внешних источников	–
Сцена 5	Выход	Из внешних источников	–

Таблица 20 – Фрейм-ситуация «Заказ Петра»

ЗАКАЗ ПЕТРА (АКО «ЗАКАЗ»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
Перечень блюд	Отбивная, темный квас	Из внешних источников	–
Перечень цен	25; 7,5	Присоединенная процедура	IF-ADDED (изменяет слот «Перечень цен»)
Сумма заказа	32,5	Присоединенная процедура	IF-ADDED (изменяет слот «Сумма заказа»)
Принял заказ	Сергей	Из внешних источников	–
Сделал заказ	Петр	Из внешних источников	–

Таблица 21 – Фрейм-ситуация «Оплата Петра»

ОПЛАТА ПЕТРА (АКО «ОПЛАТА»)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
1	**2**	**3**	**4**
Вид платежа	Наличные	Из внешних источников	IF-ADDED (изменяет слот «Чаевые»)

Продолжение таблицы 21

1	2	3	4
Чаевые	2,3	Присоединенная процедура	–
Оплатил	Фрейм-образец	Присоединенная процедура	–
Заказ	Фрейм-образец	Из внешних источников	IF-ADDED (изменяет слот «Оплатил»)

Взаимосвязь различных видов фреймов отображается графически в виде графа (рисунок 5).

Использование фреймовой модели аналогично использованию семантической, только в процессе получения ответа, кроме вершин, учитываются и слоты. Например, получить ответ на вопрос «Кто работает официантом в ресторане “Вкусная еда”?» можно следующим образом: из запроса понятно, что необходимо найти фрейм «Ресторан “Вкусная еда”» и проследить связь с фреймом «Сергей», являющимся наследником фрейма «Официант». Также можно найти слот «Место работы» и, проверив его значение во фреймах-наследниках фрейма «Официант», определить, что официантом в ресторане «Вкусная еда» работает Сергей.

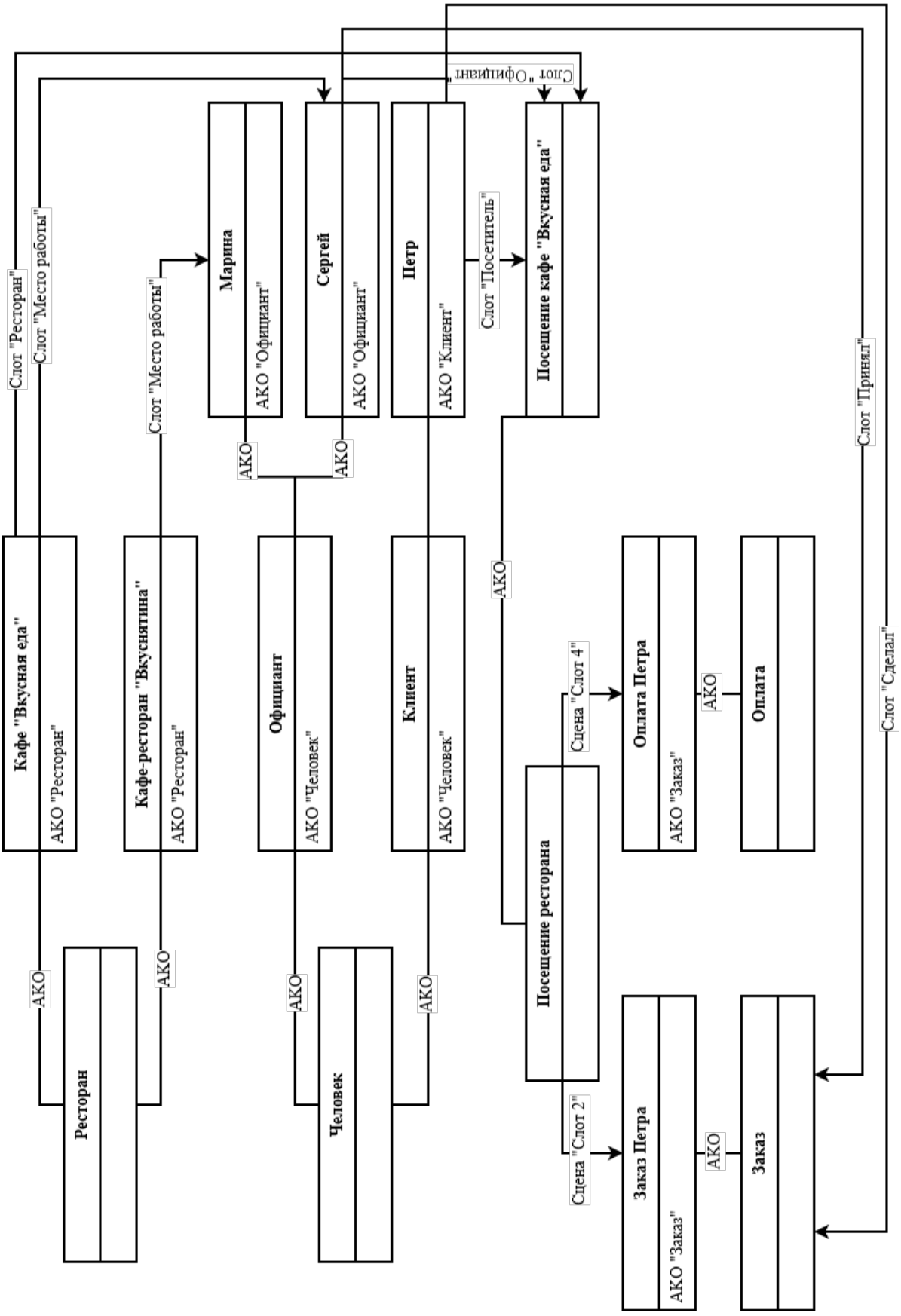

Рисунок 5 – Схема фреймов для предметной области «Ресторан»

Практическое задание № 4. Представление знаний в виде продукционной модели

Цель: научиться представлять знания с помощью продукционных моделей.

При изучении какого-либо объекта выделяют одно или несколько его свойств, совокупность которых составляет сущность этого объекта в данном рассмотрении. Для описания объекта или его отдельных свойств выбираются некоторые характеристики – величины, которые могут принимать либо количественные, либо качественные значения. В первом случае характеристики называются ***параметрами***, а во втором – ***признаками описываемого объекта***.

Каждая характеристика любого объекта может принимать значения из некоторого списка (множества) разрешенных значений (таблица 22).

Таблица 22 – Примеры фактов в виде пар «характеристика – значение»

Суждение эксперта	Объект	Пара «характеристика – значение»
«У пациента температура»	Пациент	наличие_температуры=да
«Температура высокая»	Пациент	температура=высокая
«У пациента лихорадка»	Пациент	наличие_лихорадки=да
«Цена минимальная»	Лекарство	цена=минимальная

Совокупность всех характеристик некоторого объекта (или предметной области в целом) образует так называемый список разрешенных характеристик данного объекта (предметной области). Списки разрешенных характеристик и разрешенных значений этих характеристик охватывают множество всех имеющихся фактов, подлежащих хранению в базе знаний экспертной системы. Каждый из списков не является жестко фиксированным, а может изменяться в ходе перепроектирования базы знаний, например вследствие пополнения ее новыми знаниями.

Продукция – это предложение-образец вида «Если, то», по которому осуществляется поиск в базе знаний [3, 4].

В продукции выделяют левую часть (начинается с «если» и заканчивается перед «то») и правую (начинается после «то»). Левая часть продукции – ***антецедент*** – условие выполнения правой часть продукции. Правая часть – ***консеквент*** – действие, выполняемое в случае нахождения элементов, удовлетворяющих левой части. Действие может быть промежуточным

и выступать затем в качестве консеквента или целевым, завершающим процедуру вывода.

Антецедент формируется из фактов, входных данных задачи и логических связок (И, ИЛИ, НЕ). ***Консеквент*** может представлять из себя действие по изменению фактов, данных, рекомендацию, решение задачи. Кроме этого, любая продукция имеет имя и приоритет, определяющий последовательность проверки продукций машиной вывода.

Продукции отражают причинно-следственные связи, которые и позволяют человеку принимать решения, базируясь на знаниях и предположениях о том, что есть и что будет, если что-то сделать (таблица 23).

Таблица 23 – Пример правил в виде продукций

Предпосылка (антецедент)	**Заключение (консеквент)**
ЕСЛИ температура = высокая	ТО наличие лихорадки = да
ЕСЛИ класс = млекопитающие И потребление мяса = да	ТО отряд = хищник

В процессе консультации (в процессе общения пользователя с системой) экспертная система строит дерево вывода и хранит его в памяти в некоторой внутренней форме. Успешному применению правила соответствует добавление узла с его именем, потомками которого являются узлы, соответствующие некоторым из уже выведенных фактов, а предком – новый узел, соответствующий факту, содержащемуся в правой части правила (рисунок 6).

При разработке экспертной системы необходимо понимать, что такая система должна быть ориентирована на строго конкретную предметную область, иначе она будет бесполезна.

Рассмотрим такую ситуацию: во время движения у автомобиля перегревается двигатель. Как отреагирует на это водитель? Конечно, он занервничает.

Сформулируем задачу немного иначе, в более общем виде: имеет место ситуация (*перегрев двигателя*), требуется предсказать ее последствия (*заглохнет мотор?*).

Рисунок 6 – Пример дерева вывода

Итак, прежде всего зафиксировано возникновение определенного состояния (*перегрев двигателя*), а затем в работу включаются относящиеся к нему правила:

Правило 1:
ЕСЛИ двигатель перегрелся, ТО мотор заглохнет.
Правило 2:
ЕСЛИ мотор заглохнет, ТО это приведет к денежным затратам и позднему возвращению домой.

При выполнении правила 1, мы можем прийти логически к правилу 2 и таким образом получаем прямую цепочку рассуждения.

Программно система должна будет узнать у пользователя данные о возникшей ситуации (*перегрев двигателя*). И затем, обращаясь к описанным правилам в базе знаний, должна будет спрогнозировать последующие ситуации (и/или действия).

При **прямой цепочке рассуждения** нам известны условия, **при обратной** же цепочке рассуждений известен результат и необходимо найти вызвавшие его причины.

Предположим, что автомобиль не удалось стронуть с места. В чем причина – сел аккумулятор или неисправен стартер?

Рассмотрим задачу в более общем виде. Получаем: по известному результату (*автомобиль не тронулся с места*) необходимо определить условия, которые к нему привели, т. е. по симптомам найти причину.

Правило 1:

ЕСЛИ автомобиль не заводится И сел аккумулятор, ТО не подается ток в стартер.

Правило 2:

ЕСЛИ не подается ток в стартер, ТО автомобиль не тронется с места.

В данном случае рассуждения начинаем со второго правила. При этом обратная цепочка рассуждений всегда начинается со следствия.

Задание. Необходимо построить продукционную модель для выбранной предметной области.

Описание процесса решения. Для построения продукционной модели представления знаний необходимо выполнить следующие шаги.

1 Определить целевые действия задачи (являющиеся решениями).

2 Определить промежуточные действия, или цепочку действий, между начальным состоянием и конечным (между тем, что имеется, и целевым действием).

3 Опередить условия для каждого действия, при котором его целесообразно и возможно выполнить. Определить порядок выполнения действий.

4 Добавить конкретики при необходимости, исходя из поставленной задачи.

5 Преобразовать полученный порядок действий и соответствующие им условия в продукции.

6 Для проверки правильности построения продукций записать цепочки продукций, явно проследив связи между ними – построить дерево вывода (граф).

Этот набор шагов предполагает движение при построении продукционной модели от результата к начальному состоянию, но возможно и движение от начального состояния к результату (шаги 1 и 2).

Пример решения задачи

Задача. Построить продукционную модель представления знаний *в предметной области «Ресторан» (посещение ресторана)*.

Решение.

1 Обязательное действие, выполняемое в ресторанах – поглощение пищи и ее оплата. Значит, уже есть два целевых действия «*Съесть пищу*» и «*Оплатить*», которые взаимосвязаны и следуют друг за другом.

2 Прежде чем что-либо съесть в ресторане, туда нужно прийти, дождаться официанта и сделать заказ. Кроме того, нужно выбрать, в какой именно ресторан пойти. Значит, цепочка промежуточных действий: «*Выбрать ресторан и дойти туда*», «*Сделать заказ официанту*».

3 Прежде чем идти в ресторан, необходимо убедиться, что есть необходимая сумма денег. Выбор ресторана может обуславливаться многими причинами, выберем территориальный признак – к какому ближе, в тот и идем. В разных ресторанах работают разные люди, поэтому в зависимости от выбранного ресторана официанты будут разные. Кроме того, разные рестораны специализируются на разных кухнях, поэтому заказанные блюда будут в разных ресторанах различаться. Значит, вначале идут действия, позволяющие выбрать ресторан, затем характеризуемые рестораны, а уже после – заказ, еда и оплата заказа.

4 Пусть в задаче будут рассматриваться два ресторана: «*Вкусная еда*» и «*Вкуснятина*». Первый – кафе, и заказы приносят быстрее, чем во втором, второй – кафе-ресторан. В первом работает официант Сергей, а во втором официантка Марина. Петр – это клиент.

5 Выше описанное можно преобразовать в следующие предложения типа «Если, то»:

(1) Если субъект хочет есть и у субъекта есть достаточная сумма денег, то субъект может пойти в ресторан.

(2) Если субъект ближе к ресторану «Вкусная еда», чем к ресторану «Вкуснятина» и субъект может пойти в ресторан, то субъект идет в ресторан «Вкусная еда».

(3) Если субъект ближе к ресторану «Вкуснятина», чем к ресторану «Вкусная еда» и субъект может пойти в ресторан, то субъект идет в ресторан «Вкуснятина».

(4) Если субъект идет в ресторан «Вкуснятина» и в ресторане «Вкуснятина» работает официант Марина, то у субъекта принимает заказ Марина.

(5) Если субъект идет в ресторан «Вкусная еда» и в ресторане «Вкусная еда» работает официант Сергей, то у субъекта принимает заказ Сергей.

(6) Если субъект выбрал блюда и у субъекта принимает заказ Марина, то заказ принесут через 20 мин.

(7) Если субъект выбрал блюда и у субъекта принимает заказ Сергей, то заказ принесут через 10 мин.

(8) Если заказ принесут через 20 мин или заказ принесут через 10 мин, то субъект может есть.

(9) Если субъект может есть, то после еды субъект должен оплатить заказ.

Введем обозначения для фактов (Ф), действий (Д) и продукций (П), тогда:

− Субъект = Петр;

− Ф1= субъект хочет есть;

− Ф2= у субъекта есть достаточная сумма денег;

− Ф3= субъект ближе к ресторану «Вкусная еда», чем к ресторану «Вкуснятина»;

− Ф4= в ресторане «Вкуснятина» работает официант Марина;

− Ф5= в ресторане «Вкусная еда» работает официант Сергей;

− Ф6= субъект выбрал блюда;

− Д1= субъект может пойти в ресторан;

− Д2= субъект идет в ресторан «Вкусная еда»;

− Д3= субъект идет в ресторан «Вкуснятина»;

− Д4= у субъекта принимает заказ Марина;

− Д5= у субъекта принимает заказ Сергей;

− Д6= заказ принесут через 20 мин;

− Д7= заказ принесут через 10 мин;

− Д8= после еды субъект должен оплатить заказ.

Для продукций установим приоритет (в скобках перед запятой, чем выше приоритет, тем раньше проверяется правило):

− П1 (4, Ф1 и Ф2)= Д1;

− П2 (5, Ф3 и Д1)= Д2;

− П3 (4, не Ф3 и Д1)= Д3;

− П4 (3, Д3 и Ф4)= Д4;

− П5 (3, Д2 и Ф5)= Д5;

− П6 (2, Д4)= Д6;

− П7 (2, Д5)= Д7;

− П8 (1, Д6 или Д7)= Д8.

6 Для отображения взаимосвязи продукций построим граф решения (рисунок 7).

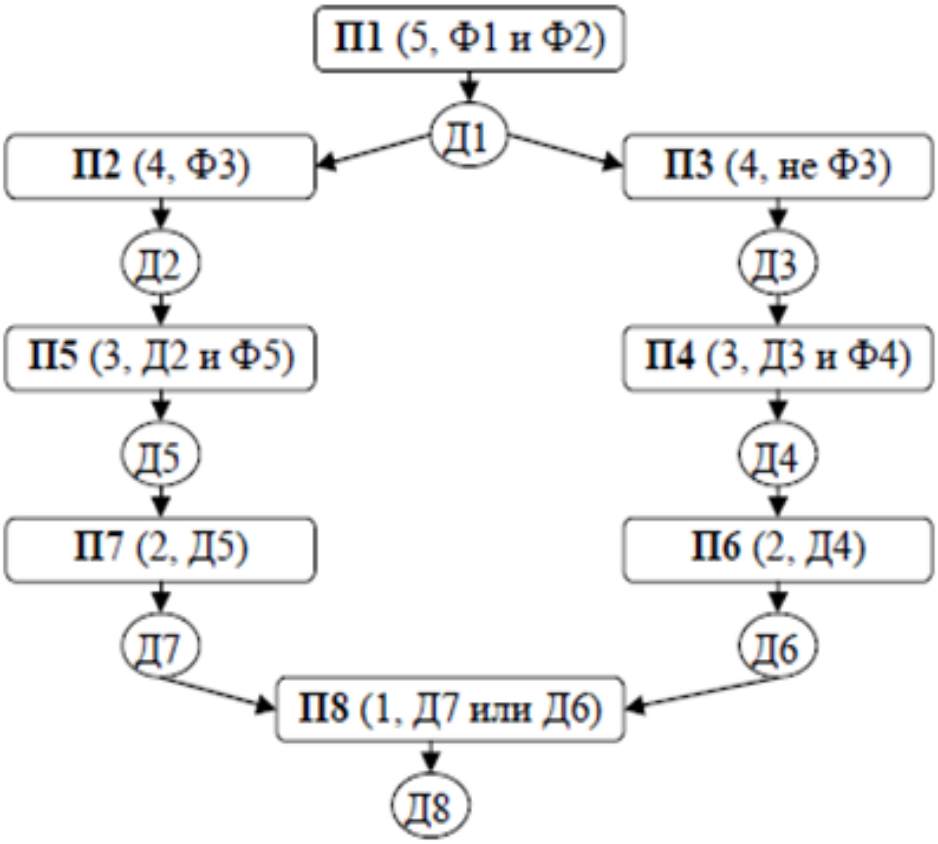

Рисунок 7 – Схема продукций предметной области «Ресторан»

2 Инструментальные средства проектирования интеллектуальных систем

Инструментальные средства проектирования интеллектуальных систем предназначены для поддержки проектирования системы на всех этапах ее жизненного цикла.

Рассмотрим некоторые такие средства.

2.1 Ostis-система

В состав каждой ostis-системы входит база знаний, решатель задач и интерфейс.

В базе знаний описывается вся информация, соответствующая некоторой части действительного мира, а также, например, знания о самой системе (ее документация) и многое другое.

База знаний – совокупность знаний, хранимых в памяти интеллектуальной компьютерной системы и достаточных для того, чтобы указанная система удовлетворяла соответствующим предъявляемым к ней требованиям (в частности, чтобы она имела соответствующий уровень интеллекта) [1, 2, 7].

База знаний – информация, хранимая в памяти кибернетической системы и имеющая высокий уровень качества по всем показателям и, в частности, высокий уровень [1, 2, 7]:

− семантической мощности языка представления информации, хранимой в памяти кибернетической системы (в базе знаний указанный язык должен быть универсальным);

− гибридности информации, хранимой в памяти кибернетической системы;

− многообразия видов знаний, хранимых в памяти кибернетической системы;

− формализованности информации, хранимой в памяти кибернетической системы;

− структурированности информации, хранимой в памяти кибернетической системы.

В качестве языка универсального представления знаний в базах знаний ostis-систем используется SC-код. Особенность данного языка заключается в том, что он

обеспечивает единую форму записи разных видов знаний (декларативных и процедурных). *SC-код является формальным языком универсального представления знаний в памяти интеллектуальных компьютерных систем и представляет собой множество нелинейных sc-текстов*. Универсальность SC-кода обеспечивается и тем, что элементами текстов SC-кода могут быть знаки описываемых сущностей любого вида, в том числе и знаки связей между описываемыми сущностями и/или их знаками. Тексты SC-кода являются графовыми структурами расширенного вида, в которых знаки описываемых связей могут соединять не только вершины (узлы) графовой структуры, но и знаки других связей. Таким образом, на SC-коде можно записать любую информацию, в том числе информацию (программу), которая будет обрабатывать другую информацию. Данный код обладает простым синтаксисом, благодаря чему прост в понимании, представлении и обработке. Так как SC-код является графовым нелинейным формальным языком, то абстрактная память ostis-системы является графодинамической (т. е. нелинейной (графовой) и структурно перестраиваемой).

Для представления баз знаний ostis-систем используется целый ряд как универсальных языков, так и специализированных языков, как формальных языков, так и естественных языков, как внутренних языков, обеспечивающих представление информации в памяти ostis-систем, так и внешних языков, обеспечивающих представление информации, вводимой в память ostis-систем либо выводимой из этой памяти. Естественные языки используются исключительно для представления файлов, хранимых в памяти ostis-системы и формально специфицируемых в рамках базы знаний этой ostis-системы.

Для эксплуатации интеллектуальных компьютерных систем, построенных на основе SC-кода, кроме способа абстрактного внутреннего представления баз знаний (SC-кода), потребуются несколько способов внешнего изображения абстрактных sc-текстов, удобных для пользователей и используемых при оформлении исходных текстов баз знаний указанных интеллектуальных компьютерных систем и исходных текстов фрагментов этих баз знаний, а также используемых для отображения пользователям различных фрагментов баз знаний по пользовательским запросам. В качестве таких способов внешнего отображения sc-текстов и предлагаются указанные выше внешние языки ostis-систем (SCg-код, SCs-код и SCn-код).

База знаний каждой ostis-системы представляет собой иерархическую систему разделов, к которым должны привязываться исходные тексты каждой новой информации, вводимой в базу знаний, и, прежде всего, исходные тексты достаточно крупных фрагментов баз знаний. *База знаний ostis-системы имеет достаточно развитую иерархическую структуру. База знаний делится на разделы. Разделы бывают атомарными и неатомарными. Неатомарный раздел состоит из сегментов. Атомарные разделы не имеют сегментов. Разделы базы знаний ostis-системы могут иметь самое различное назначение.*

Знание, описываемое в базе знаний, – дискретная информационная конструкция, представленная в некотором (дополнительно уточняемом) языке. ***Sc-знание*** – знание, представленное на SC-коде. Не каждый sc-текст является sc-знанием. В отличие от sc-текстов в знаниях важна семантическая целостность (полнота) информационных конструкций. Так, например, логическая формула со свободными переменными не является знанием. Но полный текст высказывания, включающий в себя все компоненты всех логических связок (вплоть до атомарных логических формул), является знанием. Второй пример: sc-текст, в состав которого входит sc-коннектор или связка, но не входят все компоненты этого sc-коннектора или связки, знанием не является.

Качество системы в большей части зависит от качества формализованной и структурированной базы знаний. Так, высокий уровень обучаемости ostis-систем достигается благодаря высокому уровню:

− семантической гибкости информации, хранимой в памяти ostis-системы, поскольку каждое удаление или добавление синтаксически элементарного фрагмента хранимой информации, а также удаление или добавление каждой связи инцидентности между такими элементами имеет четкую семантическую интерпретацию;

− стратифицированности хранимой информации, что обеспечивается онтологически ориентированной структуризацией базы знаний ostis-системы;

− рефлексии ostis-системы, что обеспечивается мощными метаязыковыми возможностями языка внутреннего представления информации (знаний) в памяти ostis-систем.

Преимущества ostis-систем обеспечиваются достоинствами:

− SC-кода – языка внутреннего кодирования информации, хранимой в памяти ostis-систем;

− организации sc-памяти – памяти ostis-систем;

− sc-моделей баз знаний ostis-систем – средствами структуризации таких баз знаний;

− sc-моделей решения задач – агентно-ориентированных моделей решения задач, используемых в ostis-системах.

В формализации вы часто будете сталкиваться с таким понятием, как множество. Под ***множеством*** понимается соединение в некое целое M определенных хорошо различимых предметов m нашего созерцания или нашего мышления (которые будут называться элементами множества M). ***Множество*** – мысленная сущность, которая связывает одну или несколько сущностей в целое. Более формально ***множество*** – это абстрактный математический объект, состоящий из элементов. Связь множеств с их элементами задается бинарным ориентированным отношением ***«принадлежность*»***.

Множество может быть полностью задано следующими тремя способами:

− путем перечисления (явного указания) всех его элементов (очевидно, что таким способом можно задать только конечное множество);

− с помощью определяющего высказывания, содержащего описание общего характеристического свойства, которым обладают все те и только те объекты, которые являются элементами задаваемого множества (т. е. принадлежат ему);

− с помощью теоретико-множественных операций, позволяющих однозначно задавать новые множества на основе уже заданных (это операции объединения, пересечения, разности множеств и др.).

2.2 О системе NIKA

NIKA *– интеллектуальная обучающая экспертная система, интеллектуальный ассистент, управляемый знаниями, способный вести диалог с пользователем на заданную тему или множество тем (**N**IKA is an Intelligent Knowledge-driven Assistant)* [8]. NIKA является ostis-системой, спроектированной по принципам, лежащим в основе Технологии OSTIS. Она состоит из гибридной базы знаний, гибридного решателя задач и интеллектуального интерфейса.

В базе знаний системы NIKA описываются: ее документация; предметные области, содержащие основные понятия, необходимые при решении логических задач и задач управления диалогом с пользователем; логические правила, по которым строится диалог с пользователем; агенты управления диалогом и их формальная спецификация; а также все прикладные предметные области, на тему которых может вестись диалог с пользователем.

Решатель задач и интерфейс системы NIKA будут рассмотрены далее в рамках лабораторных работ.

NIKA может отвечать на самые различные вопросы, диалог с ней можно вести на *русском языке*. **Документация по системе** доступна на русском и английском языках [8]. Документацию по платформе можно найти на онлайн-сервисе GitHub [9], она постоянно актуализируется в соответствии с изменениями платформы [10].

2.2.1 Примеры формализованных фрагментов

В базе знаний NIKA хранятся полезные шаблоны (на SCg и SCs), которые можно использовать либо для ознакомления, либо в качестве основы для построения своих конкретных фрагментов базы знаний. Расположены шаблоны в папке pattern: ***nika/kb/pattern***. В данной папке есть три модуля – *science, gwf, scs*.

В папке *gwf* есть шаблоны, написанные на SCg. Их можно использовать как основу для построения своих логических формул (правил), фраз ответа, для формализации сущностей, отношений и т. д., т. е. всего того, что придется формализовать в лабораторных работах № 1–3.

Пример формализации класса с указанием его декомпозиции и определения (примечания) представлен на рисунке 8.

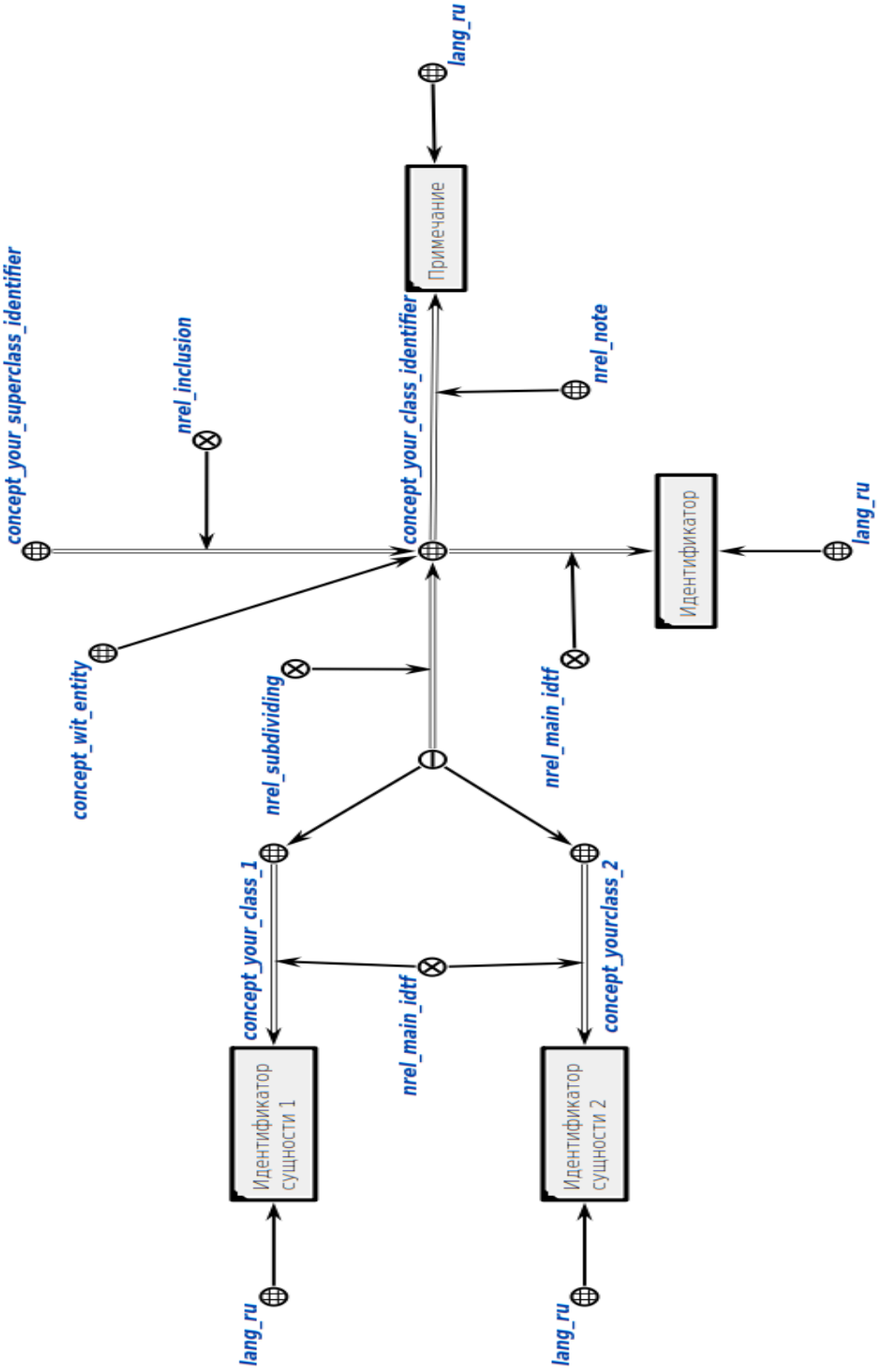

Рисунок 8 – Пример формализации класса на SCg

В папке *science* есть конкретный пример формализации сущности «science», правила классифицирования сообщения с вопросом про науку, правила ответа и фразы ответа на вопрос о науке: «Что такое наука?».

В папке *scs* представлены шаблоны, которые есть в папке *gwf*, но только на SCs. Например, ниже формализован класс, его разбиение и определение (примечание):

```
concept_your_class_identifier
<- sc_node_class;
<- concept_wit_entity;
=> nrel_main_idtf:
   [Идентификатор] (* <- lang_ru;; *);
=> nrel_note:
   [Примечание] (* <- lang_ru;; *);
<= nrel_subdividing:
   ...
   (*
     <- sc_node_tuple;;
     -> ..instance_1;;
     -> ..instance_2;;
   *);;

..instance_1
=> nrel_main_idtf:
   [Идентификатор] (* <- lang_ru;; *);;

..instance_2
=> nrel_main_idtf:
[Идентификатор] (* <- lang_ru;; *);;
```

2.2.2 Проверка правильности формализуемых фрагментов базы знаний

Для того чтобы проверить правильность формализуемых понятий, можно спросить у NIKA о сущности, которая должна быть формализована в ее базе знаний.

Например, если формализована сущность *треугольник*, то можно задать вопрос: «Что такое треугольник?». Если формализовано разбиение треугольника (его декомпозиция), то можно задать вопрос: «Каким бывает треугольник?». Примеры того, как правильно формализовать сущности, их разбиение и т. д., приведены в пункте 2.2, где указаны примеры формализации.

Примечание – Если NIKA не отвечает на вопрос «Что такое …?», то в данном случае это означает, что нужно ее обучить с помощью Wit.ai (см. с. 86, лабораторная работа № 2).

Приложение Wit.ai [11] используется для обработки нативной речи при создании программных продуктов. Данное приложение анализирует входящие сообщения (как голосовые, так и текстовые) и создает на их основе объекты.

Подробнее информацию о вопросах, на которые умеет отвечать NIKA, а также применение Wit.ai, можно найти на с. 81 (лабораторная работа № 2) в пункте «Вопросы, на которые умеет отвечать NIKA». Там представлены

примеры фрагментов, необходимых для того, чтобы NIKA отвечала на соответствующий вопрос.

Общие рекомендации по выполнению лабораторных работ

1 В результате выполнения лабораторной работы № 1 должна получиться предметная область, на тему которой в следующих лабораторных работах необходимо будет строить диалог с системой. При формализации описаний понятий и экземпляров этих понятий старайтесь думать о перспективности вашего решения. Соблюдайте вариативность определений понятий, задавайте больше связей между сущностями, старайтесь описывать несколько сущностей одного класса – это поможет вам сделать диалог с системой более интеллектуальным и разнообразным.

2 Думайте не только о вашем решении. Кроме вашей предметной области другими коллегами могут быть формализованы другие предметные области, возможно даже схожие с вашей. Проблема интеграции решений различного класса заключается вовсе не в сложности реализаций этих решений, а прежде всего в договороспособности экспертов. Всегда оставайтесь в курсе новостей о существующих решениях. Ваша задача интегрировать и переиспользовать имеющийся опыт, а не повторять его.

3 Творчески относитесь к задачам любой лабораторной работы. Лабораторная работа – это творческая задача, для которой решение придумываете вы сами. Лабораторная работа стимулирует развитие ваших творческих навыков. В это же время помните, что лабораторная работа хоть и создает творческую атмосферу, но в то же время определяет ее границы.

Выбор варианта задания. Критерии аттестации

Примерный перечень вариантов задания

(1) Транспорт
(2) Активный отдых
(3) Путешествия
(4) Рестораны
(5) Магазины
(6) Наука
(7) Косметика
(8) История
(9) Биология
(10) Искусственный интеллект
(11) Преподаватели кафедры ИИТ
(12) Журналистика
(13) Болезни
(14) Медицина
(15) Музыка
(16) Аттракционы
(17) Праздники
(18) Сказки

(19) Достопримечательности
(20) Домохозяйство
(21) Производственные товары
(22) Торговые центры
(23) Драгоценности
(24) Кондитерские изделия
(25) Мебель
(26) Языки программирования
(27) Настольные игры
(28) Механика
(29) Строительство
(30) Университет
(31) Семейные традиции
(32) Еда
(33) Одежда
(34) Животный мир
(35) Почтовые марки
(36) Спортивные олимпиады
(37) Писатели Республики Беларусь
(38) Великие математики
(39) Программное обеспечение
(40) Печатные издания
(41) Динозавры
(42) Хобби
(43) Профессии
(44) Достижения
(45) Рыбное дело
(46) Морские обитатели
(47) Мультфильмы
(48) Технические науки
(49) Спортсмены
(50) Водные ресурсы
(51) Земельные ресурсы
(52) Диеты
(53) Фитнес
(54) Здоровый образ жизни
(55) Чаи
(56) Аттракционы
(57) Фильмы ужасов
(58) Продовольствие
(59) Химические удобрения
(60) Танцы
(61) Валютно-денежные отношения
(62) Настольные игры
(63) Искусство
(64) Сыры
(65) Танки
(66) Достопримечательности Республики Беларусь
(67) Научные журналы
(68) Конференции
(69) Колбасные изделия
(70) Персоны
(71) Человекообразные
(72) Культура
(73) Этимология
(74) Квантовая физика
(75) Космос
(76) Планеты солнечной системы
(77) Созвездия
(78) Спутники
(79) Космические аппараты
(80) Астронавты
(81) Великие географические открытия
(82) Звезды Голливуда
(83) Звезды Болливуда
(84) Смысловое пространство
(85) Компьютерная техника
(86) Технологии проектирования интеллектуальных систем
(87) Интеллектуальные системы
(88) Операционные системы
(89) Аппаратное обеспечение
(90) Конные скачки
(91) Естественные языки
(92) Турниры
(93) Напитки
(94) Тайны человечества
(95) Домашние животные
(96) Комнатные растения
(97) Садовые растения
(98) Компьютерные игры
(99) Банковское дело
(100) Финансовое дело
(101) Бухгалтерский учет
(102) Кибернетические системы

(103) Системы автоматизации производства
(104) Проблемы науки
(105) Проблемы человечества
(106) Парниковый эффект
(107) Безопасность жизнедеятельности человека
(108) Зазеркалье
(109) Фокусы
(110) Охранные мероприятия
(111) Древний Рим
(112) Древняя Греция
(113) Древняя Русь
(114) Дворцы и замки
(115) Президенты
(116) Механизмы синхронизации процессов
(117) Модели памяти
(118) Нейробионика
(119) Коллективная деятельность
(120) Образовательная деятельность
(121) Технологии нового поколения
(122) Компьютеры нового поколения
(123) Семантические технологии
(124) Модели обработки информации
(125) Алгоритмы сортировки
(126) Графовые алгоритмы
(127) Религия
(128) Буддизм
(129) Православие
(130) Конфуцианство
(131) Католицизм
(132) Протестантизм
(133) Пластиковые изделия
(134) Вредные привычки
(135) Полезные привычки
(136) Деликатесы
(137) Вымершие животные
(138) Охота
(139) Фотоохота
(140) Фотографии
(141) Дизайн
(142) Разработка программного обеспечения
(143) Жизненный цикл программного обеспечения
(144) Тестирование программного обеспечения
(145) Архитектуры и паттерны проектирования программного обеспечения
(146) Робототехника
(147) Цветы
(148) Бисероплетение
(149) Керамические изделия
(150) Лепка из глины
(151) Картины
(152) Карточные игры
(153) Футбол
(154) Хоккей

Критерии аттестации:

1) структурированность понятий в предметной области (до 1 балла);

2) наполненность понятиями (до 1 балла);

3) связность экземпляров понятий (до 1,5 баллов);

4) тестируемость (до 0,5 балла);

5) качество отчета (до 1 балла);

6) умение отвечать на вопросы и решать задачи (до 3 баллов);

7) срок защиты (до 2 баллов).

Максимальный балл: 10.

Срок сдачи каждой лабораторной работы устанавливается преподавателем на первом занятии.

Алгоритм опубликования результатов работы

1 Сделать форк проекта системы NIKA [12].

2 Установить форк проекта системы у себя локально:

```
git clone https://github.com/{имя_Вашего_аккаунта}/nika
git checkout tpis-2023
```

3 Перейти в папку *kb/extra* и загрузить туда исходники фрагментов базы знаний.

4 Вернуться в корень проекта, сделать коммит:

```
git add kb/extra/
git commit -m "feat(kb): <название вашей предметной области>"
```

и пуш изменений:

```
git push
```

5 По указанию преподавателя сделать *pull-request* в Проект системы NIKA [12] (*pull-request* необходимы для получения рекомендуемой оценки для экзамена и выполняются только в случае, если лабораторная работа выполнена качественно, т. е. на 9–10 баллов, и одобрена проверяющим).

Лабораторная работа № 1. Изучение принципов работы интеллектуальных экспертных обучающих систем на примере системы NIKA, описание выбранной предметной области и фактографических высказываний для базы знаний системы NIKA

Цель: изучить структуру и принципы работы баз знаний интеллектуальных систем, освоить навыки формализации предметных областей и фактографических высказываний в базах знаний ostis-систем, научиться осуществлять навигацию по базе знаний, выявлять закономерности и противоречия в ней.

Задание

1 Выбрать тему предметной области согласно варианту (вариант выбирается по желанию, но он должен быть уникальным в рамках потока и отличаться темы курсового проектирования, также необходимо согласовать вариант с преподавателем).

2 Формализовать структуру предметной области (выделить классы объектов исследования и исследуемые отношения).

3 Описать классы объектов исследования и исследуемые отношения (суммарно не менее 30 абсолютных и относительных понятий):

− между формализуемыми понятиями нужно обязательно задавать теоретико-множественные отношения;

− для формализуемых отношений необходимо задавать свойства и домены;

− при описании понятий задавать примечания, пояснения и определения к ним;

− при формализации предметной области необходимо пользоваться текстами из открытых источников.

4 Для каждого понятия описать не менее одного экземпляра:

− полученные экземпляры связать между собой при помощи относительных понятий;

− полученная структура должна соответствовать выбранным текстам по теме предметной области.

5 Протестировать полученные фрагменты базы знаний в системе NIKA, используя графовый редактор, контекстное меню и другие возможности.

6 По результатам работы составить электронный отчет, где указать:

− цель выполненной работы;

− задание лабораторной работы;

− индивидуальный вариант;

− иллюстрацию и тестирование фрагментов базы знаний;

− выводы.

7 Ответить на вопросы и решить задачи, указанные в конце лабораторной работы.

8 Загрузить свои изменения в форк на GitHub.

Описание процесса решения

Установка проекта. Для того чтобы приступить к работе с системой NIKA, необходимо развернуть ее локально на компьютере. Проект системы NIKA располагается в репозитории на GitHub [12]. Если установлена ОС семейства Debian, то вы можете установить и развернуть систему, выполнив следующие действия:

1 Установка системы:

```
git clone -c core.longpaths=true -c core.autocrlf=true --recursive
https://github.com/ostis-apps/nika
cd nika
```

2 Запуск документации:

```
pip3 install mkdocs markdown-include mkdocs-material mkdocs-
static-i18n
mkdocs serve
```

Документацию по работе с системой NIKA также можно найти на GitHub [12].

В документации платформы, на которой спроектирована NIKA [8], можно увидеть информацию по установке и решению проблем, если они возникли в процессе установки системы. Вам понадобится страница *Build / Debian-based distros / Linux* или *Build/Docker*.

На рисунке 9 представлено диалоговое окно системы NIKA.

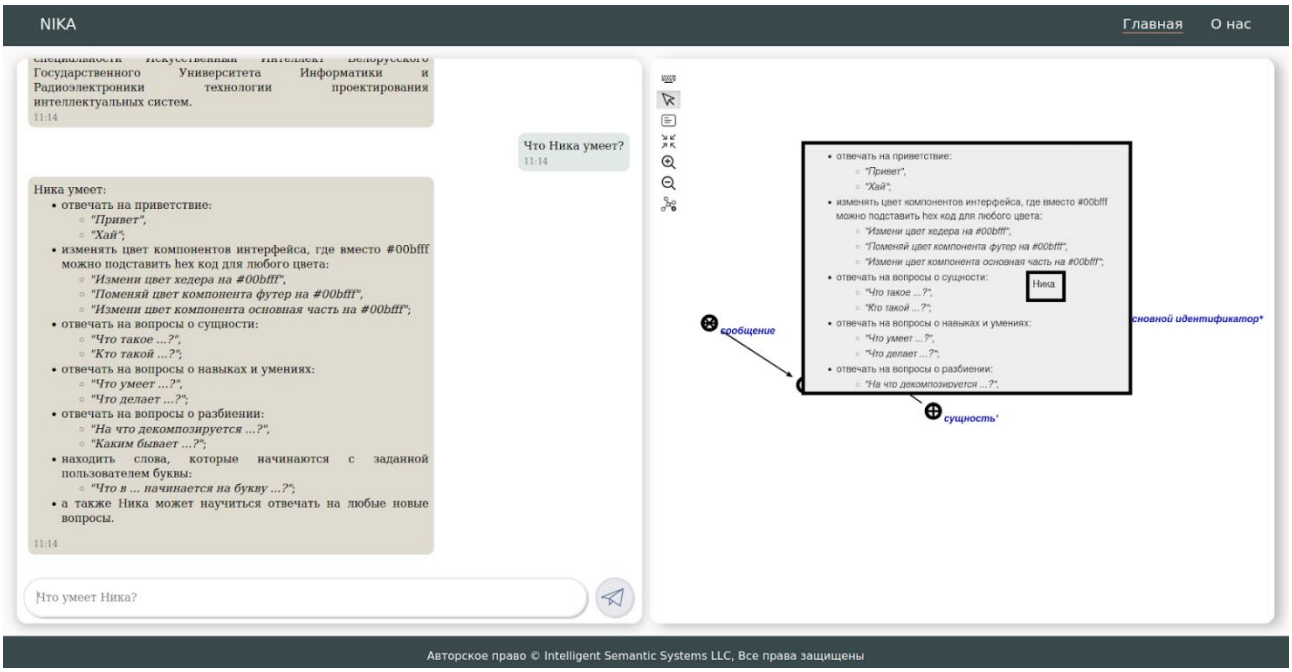

Рисунок 9 – Диалоговое окно NIKA

Переключить языки в документации [8] можно с помощью кнопки на верхней панели в правой ее части, как показано на рисунке 10.

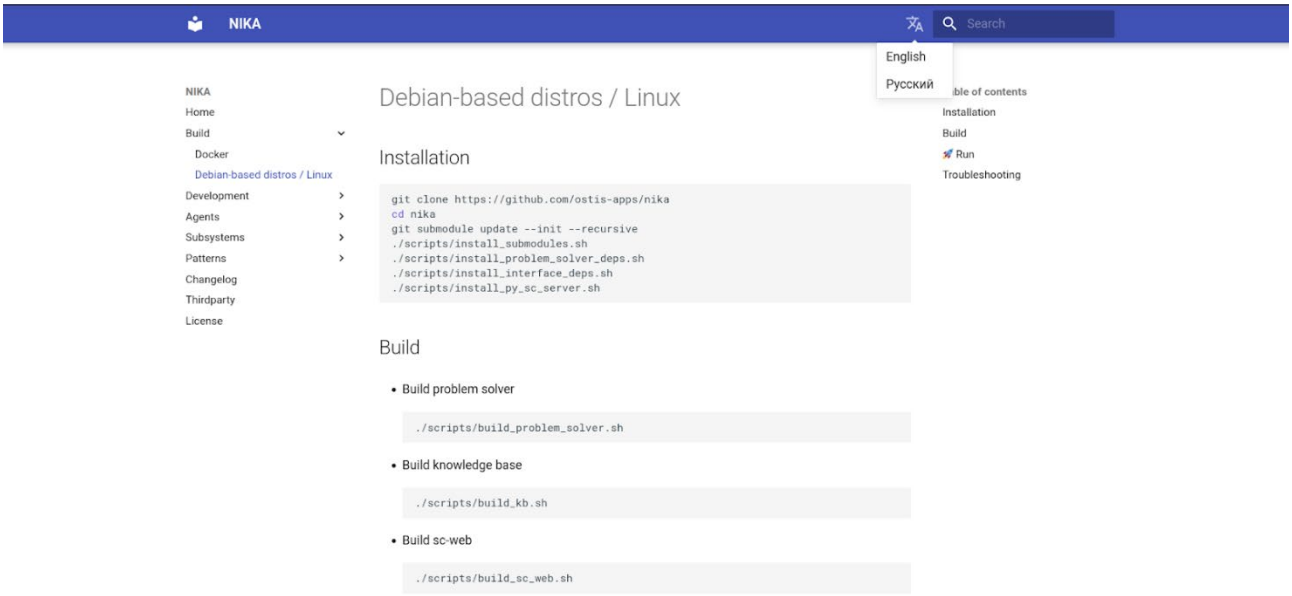

Рисунок 10 – Переключение языка в документации

Для того чтобы изучить, как работает система, можно открыть документацию в Проекте системы NIKA [12], а также документацию платформы OSTIS [13], на которой она спроектирована.

При необходимости можно настроить сервер, который запускался первой командой. Для этого можно перейти в NIKA и открыть конфигурационный файл nika.ini. Информацию о настройке sc-машины можно также найти на GitHub [14].

Средства разработки базы знаний. Для того чтобы приступить к выполнению заданий, указанных в лабораторной работе, необходимо установить графический редактор баз знаний:

− на ОС семейства Debian или MacOS необходимо использовать редактор Linux KBE [15];

− на Windows ОС, необходимо использовать редактор Windows KBE [16].

При всех вариантах архив с редактором начнет скачиваться автоматически. Для того чтобы запустить редактор, необходимо разархивировать его и запустить исполняемый файл:

− Debian или MacOS – *cd KBE & ./kbe*;

− Windows – *cd KBE 4.0 & kbe.exe*.

Кроме того, можно воспользоваться любым текстовым редактором, с помощью которого можно сохранять файлы. В текстовом редакторе можно описывать фрагменты базы знаний в строковом формате (на языке SCs).

После успешной установки всех необходимых средств можно приступать к выполнению заданий, указанных в лабораторной работе.

Пример выполнения заданий

1 Допустим, тема лабораторной работы «Интеллектуальные обучающие системы». Тогда название формализуемой предметной области и соответствующего ей раздела в базе знаний – «Предметная область интеллектуальных обучающих систем» и «Раздел. Предметная область интеллектуальных обучающих систем».

2 В папке *nika/kb/extra* необходимо создать папку для выбранной предметной области. В нашем случае название директории будет выглядеть следующим образом: *section_subject_domain_of_intelligent_study_systems*, иерархия директорий базы знаний, соответственно:

```
kb/
   extra/
         section_subject_domain_of_intelligent_study_systems/
```

3 Далее создаем исходный файл с расширением *.scs* с аналогичным названием *section_subject_domain_of_intelligent_study_systems.scs*, в котором можно начинать описывать структуру предметной области и соответствующего ей раздела:

```
/* Пример того, каким образом можно оставлять комментарии в
исходных файлах базы знаний. Очень важно отметить, что названия
sc-элементов должны быть написаны на латинице и соответствовать
требованиям, предъявляемым к синтаксису внешних знаков в SC-коде.
Со всеми требованиями можно ознакомиться в Стандарте Технологии
проектирования интеллектуальных систем OSTIS.*/
section_subject_domain_of_intelligent_study_systems
=> nrel_main_idtf:
   [Раздел. Предметная область интеллектуальных обучающих систем]
   (* <- lang_ru;; *);
   [Section. Subject domain of intelligent study systems]
   (* <- lang_en;; *);
<- atomic_section;;

/* Структура предметной области (классы объектов исследования и
исследуемые отношения)*/
```

```
section_subject_domain_of_intelligent_study_systems      =      [*
subject_domain_of_intelligent_study_systems
<- sc_node_struct;
=> nrel_main_idtf:
   [Предметная область интеллектуальных обучающих систем]
   (* <- lang_ru;; *);
   [Subject domain of intelligent study systems]
   (* <- lang_en;; *);
<- subject_domain;
-> rrel_maximum_studied_object_class:
   concept_intelligent_study_system;
   concept_intelligent_study_ostis_system;
-> rrel_not_maximum_studied_object_class:
   concept_lab_work;
-> rrel_explored_relation:
   nrel_knowledge_base;
   nrel_aim;
   nrel_lab_workshop;
   nrel_lab_work_condition;;
*];;

section_subject_domain_of_intelligent_study_systems
-> rrel_key_sc_element:
   subject_domain_of_intelligent_study_systems;
   concept_intelligent_study_system;
   concept_intelligent_study_ostis_system;
   nrel_knowledge_base;
   nrel_aim;
   nrel_lab_workshop;
   nrel_lab_work_condition;
   concept_lab_work;;
```

Примечание – Приведенная в качестве примера предметная область описана недостаточно. При выполнении заданий необходимо соблюдать требования, указанные в лабораторной работе.

Формально, ***предметная область*** – это результат интеграции (объединения) частичных семантических окрестностей, описывающих все исследуемые сущности заданного класса и имеющих одинаковые (общие) исследования (т. е. один и тот же набор отношений, которым должны

принадлежать связки, входящие в состав интегрируемых семантических окрестностей). В спецификации любой предметной области указываются максимальный класс объектов исследования (через связку ролевого отношения *rrel_maximum_studied_object_class*), немаксимальные классы объектов исследования (через связки ролевого отношения *rrel_not_maximum_studied_object_class*) и исследуемые отношения (через связки ролевого отношения *rrel_explored_relation*). Важно понимать *различие между максимальным классом объектов исследования и просто классом объектов исследования* – для максимального класса объектов исследования в заданной предметной области отсутствует другой класс объектов исследования, который был бы его надмножеством.

Основываясь на всем сказанном выше, ***предметная область*** – структура, в состав которой входят:

− основные исследуемые (описываемые) объекты – первичные и вторичные;

− различные классы исследуемых объектов;

− различные связки, компонентами которых являются исследуемые объекты (как первичные, так и вторичные), а также, возможно, другие такие связки, т. е. связки (как и объекты исследования) могут иметь различный структурный уровень;

− различные классы указанных выше связок (т. е. отношения);

− различные классы объектов, не являющихся ни объектами исследования, ни указанными выше связками, но являющихся компонентами этих связок.

При этом все классы, объявленные исследуемыми понятиями, должны быть полностью представлены в рамках данной предметной области вместе со своими элементами, элементами элементов и т. д., вплоть до терминальных элементов.

Можно говорить о типологии *предметных областей по разным структурным признакам*:

− наличие метасвязей;

− наличие исследуемых структур, входящих в состав предметной области;

− наличие исследуемых (смежных, дополнительных) объектов, которые исследуются в других предметных областях;

Каждой предметной области можно поставить в соответствие:

− семейство соответствующих ей онтологий разного вида;

− некий язык (в нашем случае – язык, построенный на основе SC-кода), тексты которого представляют различные фрагменты соответствующей предметной области.

Для эффективной коллективной разработки и эксплуатации базы знаний ostis-системы важна не просто ее структуризация, а такая структуризация, которая носит максимально объективный характер, имеет четкую семантическую интерпретацию и позволяет на основе семантических связей между структурно выделяемыми фрагментами базы знаний легко определять (локализовывать) расположение либо искомых знаний, либо новых знаний, вводимых в состав базы знаний. Такая семантическая структуризация базы знаний, формирование системы семантически связанных между собой «семантических полочек», на которых размещаются конкретные знания, удовлетворяющие четко заданным требованиям, существенно упрощает навигацию по базе знаний и четко локализует эволюцию знаний, находящихся на каждой «семантической полочке». В основе указанной семантической структуризации базы знаний лежит иерархическая система предметных областей.

С содержательной точки зрения ***предметная область представляет собой совокупность фактографических высказываний, описывающих все элементы заданного множества объектов исследования с помощью заданного набора отношений и параметров (характеристик)***. Максимальный класс указанных объектов исследования, некоторые специально выделяемые подклассы этого класса, а также указанные отношения и параметры задают систему понятий, лежащих в основе предметной области, которую будем называть схемой предметной области. Подчеркнем то, что схема предметной области, определяющая семантическую «систему координат» в рамках соответствующей предметной области, в известной степени имеет субъективный характер и является предметом соглашения (консенсуса) между специалистами в этой области. Подчеркнем также, что схема предметной области может эволюционировать (может меняться набор понятий, может уточняться их денотационная семантика).

Понятие предметной области можно считать обобщением понятия алгебраической системы, ориентированным на решение проблемы

семантической структуризации базы знаний. В памяти ostis-системы предметная область представляется обычно *бесконечной sc-структурой, которая задается (1) неким множеством исследуемых сущностей, (2) семейством отношений, алгебраических операций и параметров (свойств), заданных на множестве исследуемых сущностей, каждое из которых либо рассматривается в рамках данной предметной области, либо наследуется из другой предметной области более высокого уровня*. Каждая предметная область в базе знаний может быть представлена своим разделом. При этом на множестве таких разделов могут быть заданы не только рассмотренные выше «синтаксические» отношения, но и целый ряд семантически интерпретируемых отношений, например отношение *«быть частной предметной областью*»*. Примером связки этого отношения является связка между *предметной областью «геометрические фигуры в евклидовом пространстве»* и *предметной областью «планарные фигуры евклидова пространства»*. Таких отношений, заданных на множестве предметных областей и уточняющих характер соотношения между множествами исследуемых сущностей, а также рассматриваемыми или наследуемыми отношениями, алгебраическими операциями и параметрами, существует достаточно много. Но если к этому добавить анализ соотношения не только между самими предметными областями, но и соотношения между представляющими их sc-структурами в рамках базы знаний конкретной ostis-системы, а также в рамках глобального смыслового пространства (sc-пространства), то число семантически интерпретируемых отношений, заданных на множестве предметных областей существенно расширится. К числу важных отношений, заданных на множестве предметных областей, относятся также отношения, связывающие предметные области с различными структурно выделяемыми фрагментами базы знаний, которые предметными областями не являются.

4 После описания структуры выбранной предметной области можно приступать к описанию всех указанных понятий. Перед этим в папке *nika/kb/extra/section_subject_domain_of_intelligent_study_systems* необходимо создать папки для абсолютных и относительных понятий – *concepts* и *relations*, соответственно, где будут описываться абсолютные и относительные понятия. Иерархия директорий базы знаний, соответственно, обновится:

```
kb/
  extra/
     section_subject_domain_of_intelligent_study_systems/
       section_subject_domain_of_intelligent_study_systems.scs
        concepts/
        relations/
```

В качестве примера абсолютного понятия опишем понятие интеллектуальной обучающей системы с целью дальнейшего описания сущности самой системы NIKA. Таким образом, в следующих лабораторных работах можно будет строить диалог на тех знаниях, которые имеются о NIKA в ее базе знаний. Это могут быть такие типовые вопросы, как «Кто такая Ника?», «Что нужно сделать в лабораторной работе № 1?», «Когда нужно сдать первую лабораторную работу?» и т. д. Содержание и глубина вопросов зависит от качества формализованной вами предметной области.

```
concept_intelligent_study_system
<- sc_node_class;
<- class;
=> nrel_main_idtf:
   [интеллектуальная обучающая система]
   (* <- lang_ru;; *);
<= nrel_inclusion:
   concept_intelligent_system
   (* => nrel_main_idtf:
         [интеллектуальная система]
         (* <- lang_ru;; *);;
   *);
=> nrel_note:
   [Такого рода системы по сравнению с традиционными системами
   электронного обучения (например, электронными учебниками)
   обладают рядом существенных преимуществ.]
   (* <- lang_ru;; *);
<= nrel_inclusion:
   concept_intelligent_help_system
   (*
    => nrel_main_idtf:
       [интеллектуальная справочная система]
       (* <- lang_ru;; *);;
    => nrel_note:
```

```
        [Каждая интеллектуальная обучающая система в качестве
        простейшего средства изучения учебного материала
        предполагает наличие средств навигации по этому материалу и
        средств задания по нему различных вопросов. Системы,
        обладающие только таким ограниченным набором возможностей,
        названы интеллектуальными справочными системами. Таким
        образом, можно сказать, что интеллектуальная обучающая
        система обязательно реализует в себе функции
        интеллектуальной справочной системы.]
        (* <- lang_ru;; *);;
    *);
 => nrel_inclusion:
    concept_intelligent_study_ostis_system
    /* Иногда есть необходимость не описывать некоторые понятия,
    участвующие в описании формализуемого понятия. Это может
    быть объяснено тем, что связанное понятие принадлежит
    другой предметной области и никак не связано с этой. Но для
    дополнительной визуализации фрагментов базы знаний
    можно задавать идентификаторы на конкретном языке. */
    (* => nrel_idtf:
          [интеллектуальная обучающая система, построенная на
          основе Технологии OSTIS]
          (* <- lang_ru;; *);;
    *);;
```

Как видно, в формализованном понятии указаны идентификаторы понятий, примечания, подмножества и надмножества класса. Данное описание не является полным, его можно улучшать и дополнять. Также примеры формализации абсолютных понятий вы можете найти в Стандарте Технологии OSTIS [1] и на сайте ims.ostis.net [2].

В качестве исследуемого отношения можно описать понятие бинарного ориентированного отношения *«лабораторный практикум по системе*»*:

```
nrel_lab_workshop
<- sc_node_norole_relation;
<- relation;
<- binary_relation;
<- oriented_relation;
<- antireflexive_relation;
<- asymmetric_relation;
<- antitransitive_relation;
```

```
=> nrel_main_idtf:
   [лабораторный практикум по системе*]
   (* <- lang_ru;; *);;
=> nrel_first_domain: concept_intelligent_system;
=> nrel_second_domain: concept_lab_work;
=> nrel_definitional_domain:
   ...
   (*
    <= nrel_combination: {
       concept_intelligent_system;
       concept_lab_work
       };;
  *);
<- rrel_key_sc_element:
   ...
   (*
     => nrel_main_idtf:
        [Опр. (лабораторный практикум по системе*)]
        (* <- lang_ru;; *);
     <- definition;;
     <= nrel_sc_text_translation:
        ...
        (*
          -> rrel_example:
             [Лабораторный практикум по системе* - бинарное
             ориентированное отношение, связывающее
             интеллектуальную систему и множество лабораторных
             работ, являющихся лабораторным практикумом по этой
             системе.]
             (* <- lang_ru;; *);;
        *);;
     <= nrel_using_constants: {
        concept_intelligent_system;
        concept_lab_work
        };;
   *);;
```

Необходимо ознакомиться с используемыми при формализации понятиями. ***Понятие*** – класс сущностей, который входит в состав по крайней мере одной предметной области в качестве (в роли) ключевого исследуемого

понятия. Каждому понятию соответствует по крайней мере одна предметная область, в которой оно является исследуемым понятием и в которой рассматриваются основные характеристики этого понятия. Между понятиями можно строить разные теоретико-множественные отношения: включение одного понятия в другое, принадлежность одного понятия другому, различные теоретико-множественные операции над понятиями и т. д.

Все понятия (классы объектов исследования и исследуемые отношения), описываемые в рамках базы знаний, делятся на *абсолютные (классы)* и *относительные (отношения) понятия*.

Класс – множество элементов, обладающих какими-либо явно указываемыми общими свойствами. Конкретный элемент, принадлежащий классу, называется ***экземпляром класса***. Классом может быть, например, группа людей, занимающихся некоторой деятельностью, либо множество фруктов, обладающих схожими признаками, а экземпляром класса – конкретный человек либо фрукт, соответственно, принадлежащие этим классам.

Отношение, заданное на множестве M – это подмножество декартового произведения этого множества самого на себя n раз. В более широком смысле ***отношение*** – это математическая структура, которая формально определяет свойства различных объектов и их взаимосвязи. Отношения делятся на *бинарные* и *небинарные*, *ориентированные* и *неориентированные*, *ролевые* и *неролевые*.

Бинарное отношение – это множество таких связок на множестве M, которые являются подмножеством декартова произведения множества M самого на себя. Если бинарное отношение R задано на множестве M и два элемента этого множества a и b связаны данным отношением, то будем обозначать такую связь как aRb. Бинарные отношения могут быть *рефлексивными*, *антирефлексивными* или *частично рефлексивными*; *симметричными*, *антисимметричными* или *частично симметричными*; *транзитивными*, *антитранзитивными* или *частично транзитивными*.

Небинарное отношение – это множество связок на множестве M, хотя бы одна из связок каждого из которых имеет значение мощности больше двух. Унарное отношение не является небинарным отношением.

Ориентированное отношение – это множество таких отношений, каждая связка которых является кортежем.

Неориентированное отношение – это множество таких отношений, каждая связка которых является неориентированным множеством.

Ролевое отношение – это отношение, являющееся подмножеством отношения принадлежности. В конце каждого идентификатора, соответствующего экземплярам класса «ролевое отношение», не являющегося системным, ставится знак «’».

Например: *ключевой экземпляр’*.

Из-за ограничений в разрешенном алфавите символов в системном идентификаторе не может быть использован знак «”», поэтому в начале каждого системного идентификатора, соответствующего экземпляру класса «ролевое отношение», ставится префикс *rrel_*.

Например: *rrel_key_sc_element*.

Неролевое отношение – отношение, не являющееся подмножеством отношения принадлежности.

В конце каждого идентификатора, соответствующего экземплярам класса «неролевое отношение», не являющегося системным, ставится знак «*».

Из-за ограничений в разрешенном алфавите символов в системном идентификаторе не может быть использован знак «*», поэтому в начале каждого системного идентификатора, соответствующего экземпляру класса «неролевое отношение», ставится префикс *nrel_*.

Например: *nrel_inclusion*.

С остальными определениями понятий и примерами формализации понятий можно подробно ознакомиться в Стандарте Технологии OSTIS [1] и на сайте ims.ostis.net [2].

После описания всех абсолютных и относительных понятий ваша директория базы знаний будет выглядеть таким образом:

```
kb/
   extra/
     section_subject_domain_of_intelligent_study_systems/
       section_subject_domain_of_intelligent_study_systems.scs
       concepts/
         concept_intelligent_study_system.scs
       relations/
         nrel_lab_workshop.scs
```

При выполнении заданий лабораторной работы для каждого описываемого понятия необходимо создавать свой файл с расширением *.scs*.

5 На заключительным этапе формализации предметной области следует описать все необходимые экземпляры используемых вами понятий, а также указать связи между ними, принадлежащие как отношениям формализуемой вами предметной области, так и другим смежным предметным областям.

Проиллюстрируем эти требования на примере сущности NIKA как экземпляра абсолютного понятия «интеллектуальная обучающая система». Для этого рядом в папке *concepts/* следует создать папку *instances/*:

```
kb/
   extra/
     section_subject_domain_of_intelligent_study_systems/
      section_subject_domain_of_intelligent_study_systems.scs
       concepts/
         concept_intelligent_study_system.scs
         instances/
       relations/
         nrel_lab_workshop.scs
```

В ней можно описывать все необходимые вам экземпляры абсолютных понятий. Так, в файле *nika.scs* опишем следующие краткие сведения о системе:

```
nika
<- concept_intelligent_study_system;
<- concept_expert_system;
=> nrel_main_idtf:
   [Ника]
   (* <- lang_ru;; *);
=> nrel_aim:
   [расширить круг решаемых задач, устранить недостатки ранее
   использованного своего аналога, упростить и автоматизировать
   процесс обучения студентов, изучающих интеллектуальные
   экспертные системы.]
   (* <- lang_ru;; *);
   [обучить студентов специальности Искусственный интеллект
   Белорусского Государственного Университета Информатики и
   Радиоэлектроники технологии проектирования интеллектуальных
   систем.]
   (* <- lang_ru;; *);
=> nrel_lab_workshop:
  nika_lab_workshop
   (* => nrel_main_idtf:
```

```
        [Лабораторный практикум по системе Ника]
        (* <- lang_ru;; *);;
     <= nrel_inclusion: concept_lab_work;;
     -> rrel_1: nika_lab_work_1;;
  *);;

nika_lab_work_1
<- concept_lab_work;
<- concept_wit_entity;
=> nrel_main_idtf:
   [Лабораторная работа №1]
   (* <- lang_ru;; *);
=> nrel_idtf:
   [Лабораторная работа №1. Изучение принципов работы
   интеллектуальных экспертных обучающих систем на примере системы
   NIKA, описание выбранной предметной области и фактографических
   высказываний для базы знаний системы NIKA]
   (* <- lang_ru;; *);
=> nrel_lab_work_number:
   [№1]
   (* <- lang_ru;; *);
=> nrel_lab_work_condition:
   [1) Выбрать тему предметной области согласно варианту (вариант
   выбирается случайным образом и уникален в рамках потока);
    2) Формализовать структуру предметной области (выделить классы
    объектов исследования и исследуемые отношения);
    3) Описать классы объектов исследования и исследуемые отношения
    (суммарно не менее 30 абсолютных и относительных понятий):
    • между формализуемыми понятиями нужно обязательно задавать
    теоретико-множественные отношения;
    • для формализуемых отношений необходимо задавать свойства и
    домены;
    • при описании понятий задавать примечания, пояснения и
    определения к ним;
    • при формализации предметной области необходимо пользоваться
    текстами из открытых источников.
    4) Для каждого понятия описать не менее одного экземпляра:
    • полученные экземпляры связать между собой при помощи
    относительных понятий;
    • полученная структура должна соответствовать выбранным текстам
    по теме предметной области.
```

```
    5) Протестировать полученные фрагменты базы знаний в системе
    NIKA, используя графовый редактор, контекстное меню и другие
    возможности;
    6) По результатом работы составить отчёт, где указать:
    • цель выполненной работы,
    • задание лабораторной работы,
    • индивидуальный вариант,
    • иллюстрацию и тестирование фрагментов базы знаний
    • и выводы.
    7) Уметь отвечать на вопросы и решать задачи, указанные в конце
    лабораторной работы.
    8) После одобрения преподавателем результатов работы залить
    изменения в свой форк проекта NIKA на github.]
   (* <- lang_ru;; *);
=> nrel_deadline:
   [до 1 октября текущего года]
   (* <- lang_ru;; *);;
```

В данном фрагменте текста можно увидеть описание сущности системы NIKA, ее цели, лабораторный практикум по ней и задание по первой лабораторной работе.

После выполнения первых четырех заданий лабораторной работы должна сформироваться логически целостная часть предметной области на выбранную вами тему.

Дополнительные примеры формализации. В качестве формата файлов может быть выбран *html*. Это означает, что в ходе формализации понятий может быть использован не только текст, но и картинки, видео – все, что может быть представлено в *html*.

Пример приведен на рисунке 11.

Для того чтобы внешний файл (sc-ссылка) был распознан как файл с форматом *html*, необходимо провести бинарную ориентированную дугу от файла к узлу *format_html* и обозначить данное отношение как *nrel_format*. В данном случае картинка была вставлена с помощью *<img src=СсылкаИзИнтернета>*.

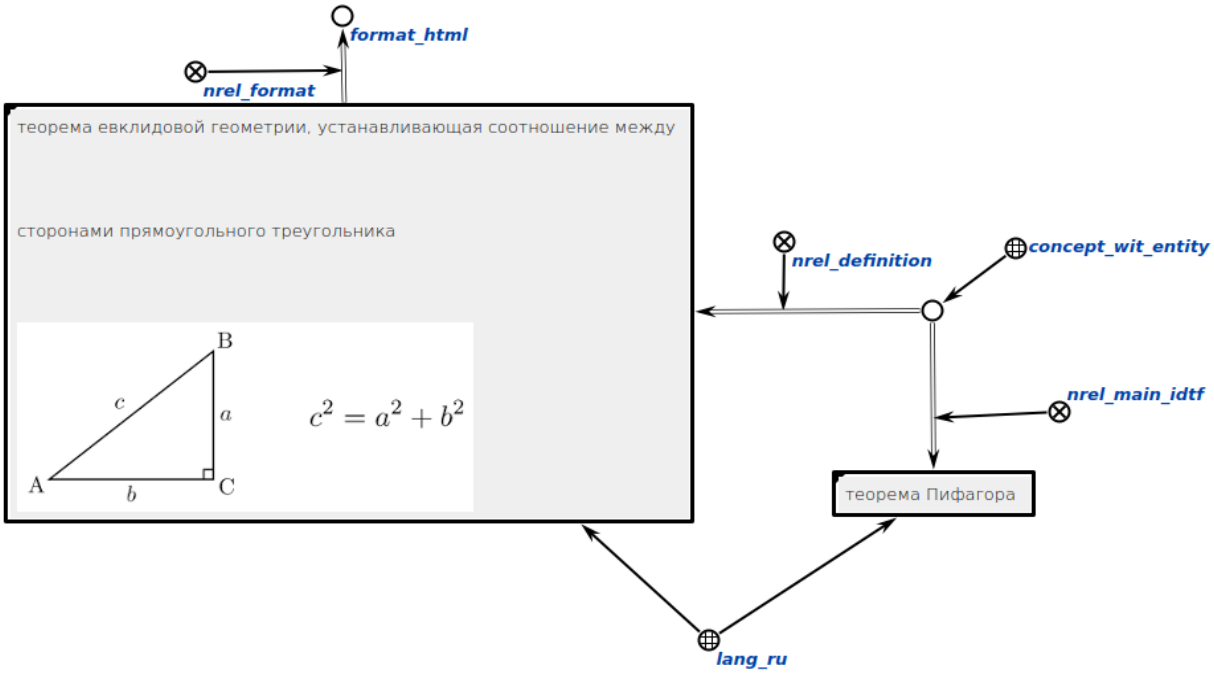

Рисунок 11 – Пример использования картинки при формализации понятий

Ссылку из интернета нужно найти самостоятельно. Используя данный фрагмент базы знаний, NIKA сможет дать соответствующий ответ (рисунок 12).

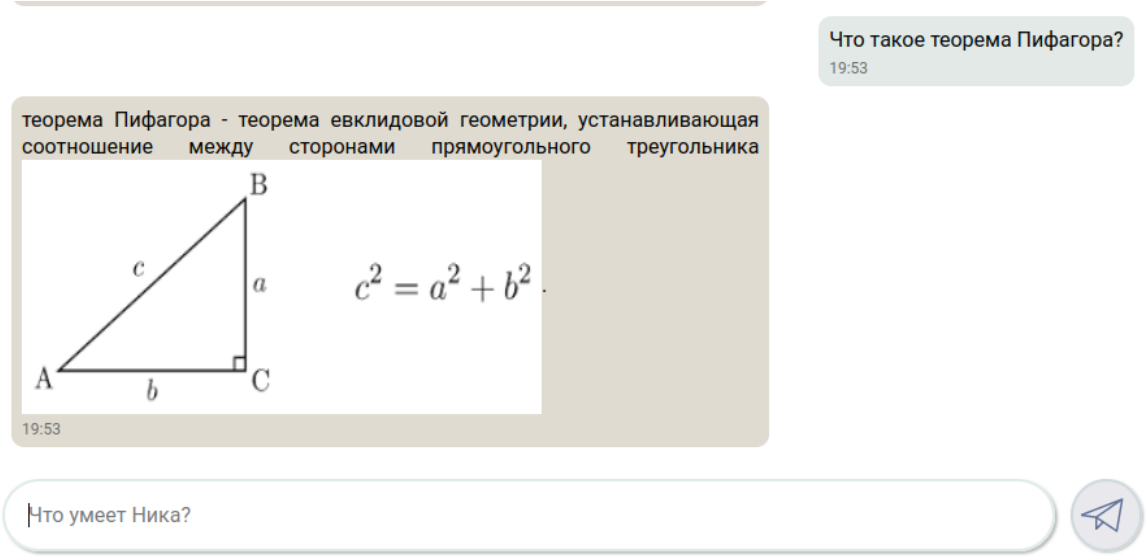

Рисунок 12 – Пример ответа системы NIKA на вопрос о теореме Пифагора

Еще один пример на SCs представлен ниже:

```
nika
<- concept_wit_entity;
=> nrel_main_idtf:
    [Ника]
    (* <- lang_ru;; *);
```

```
    [Nika]
    (* <- lang_en;; *);
=> nrel_skill:
    [
    <ul>
        <li>отвечать на приветствие:</li>
            <ul>
            <li><i>"Привет"</i>,</li>
            <li><i>"Хай"</i>;</li>
            </ul>
        <li> изменять цвет компонентов интерфейса, где вместо
        #00bfff можно подставить hex код для любого цвета:</li>
            <ul>
            <li><i>"Измени цвет хедера на #00bfff"</i>,</li>
            <li><i>"Поменяй цвет компонента футер на
            #00bfff"</i>,</li>
            <li><i>"Измени цвет компонента основная часть на
            #00bfff"</i>;</li>
            </ul>
      <li>отвечать на вопросы о сущности: </li>
          <ul>
          <li><i>"Что такое ...?"</i>,</li>
          <li><i>"Кто такой ...?"</i>;</li>
          </ul>
      <li>отвечать на вопросы о навыках и умениях:</li>
          <ul>
          <li><i>"Что умеет ...?"</i>,</li>
          <li><i>"Что делает ...?"</i>;</li>
          </ul>
      <li>отвечать на вопросы про условия и срок сдачи лабораторных
      работ, где спросить можно не только про первую лабу, но и про
      остальные три:</li>
          <ul>
          <li><i>"Когда защита первой лабы?"</i>,</li>
          <li><i>"Когда нужно сдавать первую лр?"</i>;</li>
          <li><i>"Какое условие 1 лр?"</i>,</li>
          <li><i>"Что нужно делать в лабораторной работе
          №1?"</i>;</li>
          </ul>
```

```
    <li>отвечать на вопросы о разбиении: </li>
        <ul>
        <li><i>"На что декомпозируется ...?"</i>,</li>
        <li><i>"Каким бывает ...?"</i>;</li>
        </ul>
    <li>находить слова, которые начинаются с заданной
    пользователем буквы:</li>
        <ul>
        <li><i>"Что в ... начинается на букву ...?"</i>;</li>
        </ul>
    <li> а также Ника может научиться отвечать на любые новые
    вопросы.</li>
</ul>]
(* <- lang_ru;;
   => nrel_format: format_html;;
*);;
```

Используя данный фрагмент, NIKA сможет дать ответ, представленный на рисунке 13.

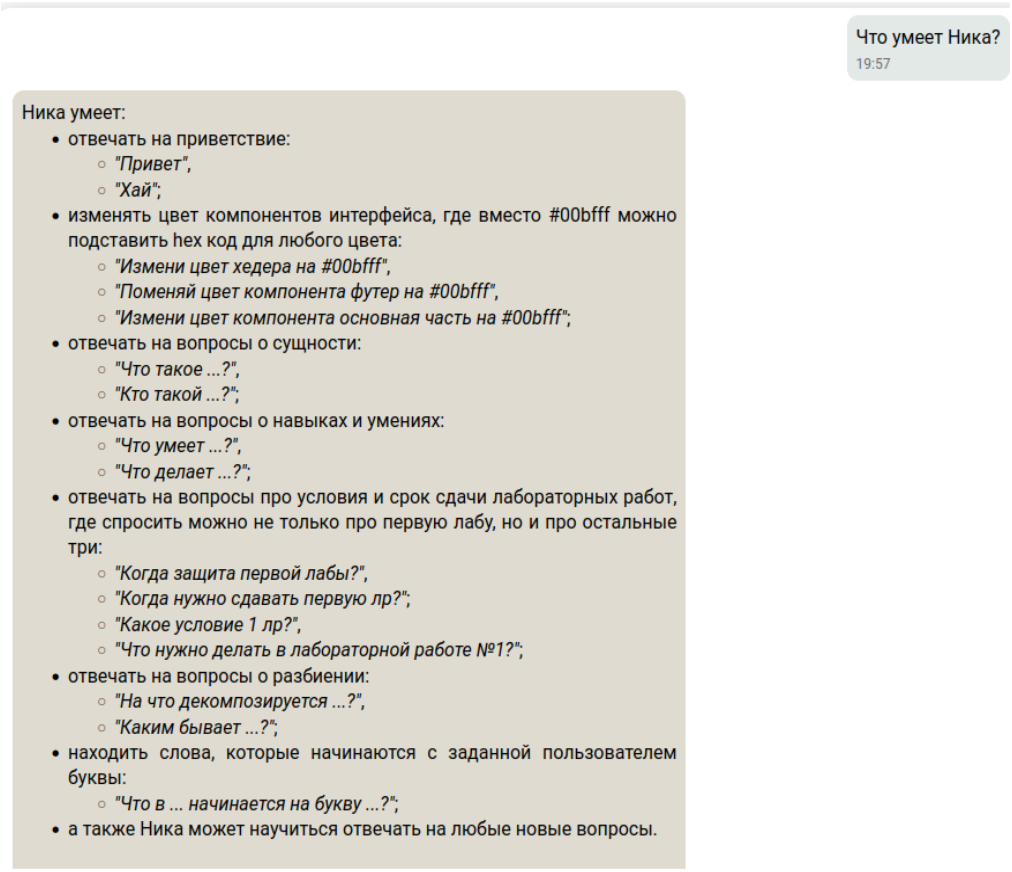

Рисунок 13 – Ответы системы NIKA о разбиении

Подготовка к защите

1 Протестировать фрагменты предметной области базы знаний в системе.

2 Сделать отчет (электронная версия) с иллюстрацией тестирования фрагментов базы знаний в системе.

3 Загрузить изменения в форк на GitHub со всеми выполненными заданиями.

4 Подготовиться к ответу на вопросы и решению задач. Повторить теоретический материал.

Ход защиты

1 Проверка отчета.

2 Иллюстрация и тестирование формализованных фрагментов базы знаний.

3 Ответы на вопросы преподавателя и решение задач.

4 Выставление оценки.

Вопросы для подготовки к защите

1 Что такое Технология проектирования интеллектуальных систем? Какие цели она ставит и какие задачи решает?

2 Какие виды интеллектуальных систем вы знаете?

3 Что является объектом изучения искусственного интеллекта? Какие направления в нем вам известны?

4 Что такое интеллектуальная компьютерная система? Какими свойствами она должна обладать?

5 В чем отличие интеллектуальной компьютерной системы от искусственной интеллектуальной системы?

6 Из каких компонентов состоит любая ostis-система? Возможно ли существование ostis-системы без какого-либо из компонентов? Аргументируйте.

7 Что такое база знаний ostis-системы? Какие знания можно представлять в ней?

8 Что такое SC-код? Какие задачи решаются с помощью него? Приведите пример фрагмент текста на этом языке.

9 Что такое знание? Приведите пример знания.

10 В чем отличие знания от информации? Приведите пример знания и информации и обоснуйте различие.

11 Какими достоинствами наделяет база знаний ostis-систему?

12 Как организуется и структурируется база знаний? Приведите пример разделов и предметных областей, формирующих ее структуру?

13 Что такое система NIKA? Какими функциями она обладает? Какие цели она преследует?

14 Что такое абсолютное и относительное понятие?

15 Объясните разницу между абсолютным и относительным понятиями и сформулируйте критерии выбора класса понятий при их формализации.

16 Из каких основных частей состоит предметная область, описываемая в рамках базы знаний?

17 В чем различие максимального класса объектов исследования и класса объектов исследования? Приведите пример.

18 Приведите пример теоретико-множественных отношений между понятиями различного вида? Зачем необходимо их указывать?

19 Какие существуют правила идентификации элементов SC-кода? Как вы считаете, какие требования часто забываются при идентификации?

20 Приведите алгоритм формализации предметной области базы знаний. На что необходимо обратить внимание в нем? Какие требования должны выполняться при формализации фрагментов базы знаний?

Задачи

1 Формализовать фактографическое высказывание:

(1) Угол AOB равен 140°. Луч OC делит угол AOB на два угла. Найдите угол, образованный биссектрисами углов AOC и COB.

(2) Разность двух углов, образованных при пересечении двух прямых, равна 24°. Найдите меньший из этих углов.

(3) Продолжения хорд AB и CD окружности с диаметром AD пересекаются под углом 25°. Найдите острый угол между хордами AC и BD.

(4) Караси – неприхотливые промысловые рыбы. Они являются ценным объектом прудового хозяйства благодаря способности выживать при низком (до 2–3 мг/л) содержании кислорода в воде. Тело высокое с толстой спиной, умеренно сжатое с боков. Чешуя крупная и гладкая на ощупь. Окраска варьируется в зависимости от места обитания.

(5) Планетарная туманность в созвездии Лиры имеет угловой диаметр 83″ и находится на расстоянии 660 пк. Каковы линейные размеры туманности в астрономических единицах?

(6) Человек – общественное существо, обладающее разумом и сознанием; субъект общественно-исторической деятельности и культуры, относящийся к виду «Человек разумный». По научным данным, возник на Земле в результате эволюционного процесса – антропогенеза, детали которого продолжают изучаться.

(7) С помощью sc-элементов можно описать любые связи между ними и/или между сущностями, которые обозначаются этими sc-элементами. При этом указанные связи трактуются как множества связываемых sc-элементов и обозначаются sc-ребрами, sc-дугами, а в случае небинарных связей – sc-узлами.

(8) Чему равен КПД источника тока с ЭДС 12 В и внутренним сопротивлением 0,5 Ом, если в цепи течет ток силой 4,8 А?

2 Формализовать часть предметной области на тему:

(1) Обитатели животного мира.

(2) Сотрудники университета.

(3) Сотрудники IT-компании.

(4) Великие географические открытия.

(5) Модели решения задач.

(6) Компьютерные системы.

(7) Правотворчество.

(8) Философские направления.

(9) Полезные ископаемые.

(10) Геометрия.

(11) Логика.

(12) История.

(13) Армия.

3 Устранить противоречия в получившемся фрагменте текста, если они есть.

Лабораторная работа № 2. Обучение классификатора сообщений Wit.ai системы NIKA

Цель: изучить принципы работы классификатора сообщений, приобрести навыки определения классов сообщений, выделения сущностей и признаков в них, формально специфицировать сценарий диалога.

Задание

1 Описать сценарий диалога по предметной области из лабораторной работы № 1. Сценарий диалога должен состоять как минимум из восьми сообщений (четыре вопроса по теме и четыре соответствующих ответа).

2 Обучить классификатор сообщений Wit.ai отвечать на вопрос «Что такое …?» для каждой формализованной сущности (см. пример на с. 86) и протестировать с помощью вопроса в диалоговом окне NIKA (в результате NIKA должна отвечать на данный вопрос для всех понятий).

3 Выделить классы и сущности сообщений, используемых в описанном сценарии диалога. Каждое сообщение должно иметь хотя бы один класс.

4 По результатам работы составить отчет, где указать цель, задание, вариант и показать тестирование сценария диалога.

5 Ответить на вопросы и решить задачи, указанные в конце данной лабораторной работы.

6 Загрузить изменения в свой форк на GitHub.

Алгоритм опубликования результатов работы

1 Сделать форк проекта системы NIKA [12].

2 Установить форк проекта системы у себя локально:

```
git clone https://github.com/{имя_вашего_аккаунта}/nika
git checkout tpis-2023
```

3 Перейти в папку *kb/extra* и загрузить туда исходники фрагментов базы знаний.

4 Также в папку *kb/extra/{папка с исходниками предметной области}/classification/* загрузить исходники с результатами классификации сообщений, предварительно сделав экспорт из Wit.ai в формате *json*.

5 Вернуться в корень проекта, сделать коммит:

```
git add kb/extra/
git commit -m "feat(kb): <название вашей предметной области>"
```

и сделать пуш изменений: *git push*.

6 Загрузить изменения в свой форк.

Теоретические сведения

Диалоговым экспертным системам недостаточно иметь только формализацию изучаемой предметной области. Системе также необходимо иметь навыки общения, механизмы понимания пользователя. Одним из таких механизмов выступает классификация сообщений пользователя. Исходя из результата классификации система определяет, как следует ответить на то или иное сообщение. ***Задача классификации*** – задача разделения множества наблюдений (объектов) на группы, называемые классами, на основе анализа их формального описания.

Классификация сообщений помогает компьютерной системе понять намерения (*intents*) пользователя, который выражает свои мысли на естественном языке, а также извлекать другую информацию из сообщения.

Сообщение можно классифицировать по теме, времени, типу и т. д. **Классификация сообщений по теме** подразумевает определение основной мысли (намерения), которую хотел передать пользователь, а также объектов (сущностей), указанных в сообщении. **Классификация сообщений по времени** позволяет выделить то, в каком времени происходят события в сообщении: сообщение о настоящем, прошедшем или будущем. **Классификация сообщений по типу** определяет цель высказывания сообщения: повествовательное, вопросительное или восклицательное сообщение. Классификация по времени и типу основывается на грамматических признаках предложения, в то время как классификация по теме предполагает обучение некоторого классификатора с использованием сообщений и указанных для них классов.

В данной лабораторной работе мы рассмотрим классификацию сообщений по теме. Например, сообщения «Сколько времени?», «Который час в Минске?», «Подскажите, пожалуйста, сколько сейчас времени в Голландии.» можно классифицировать как *сообщения запроса времени*. Как вы можете заметить, некоторые сообщения могут запрашивать время с указанием города или страны, а некоторые без этого уточнения. Для того чтобы определить ключевые элементы сообщения, выделяются сущности (*entities*) сообщения. Выделение сущностей позволяет различным образом отвечать на сообщения некоторого класса, при этом ответ на сообщение будет зависеть, например, от конкретного

города, названия спортивной команды, фильма и других сущностей. В сообщении «Который час в Минске?» можно выделить сущность *Минск*, в сообщении «Подскажите, сколько сейчас времени в Голландии.» – *Голландия*, а в сообщении «Сколько времени?» сущностей нет, в таком случае система может использовать значения по умолчанию или из других источников, например определить местоположение пользователя или извлечь знание из истории диалога.

Сообщение необязательно должно иметь только класс намерения (цели) и сущности, определяемые сообщением, также можно классифицировать сообщения по некоторым признакам (*traits*), например по эмоциональной окраске. Одну и ту же мысль можно выражать по-разному. Сообщение может быть положительной, нейтральной или негативной эмоциональной окраски. «Подскажите, сколько сейчас времени в Голландии.» можно классифицировать как сообщение с положительной эмоциональной окраской, так как оно содержит вежливые конструкции, сообщение «Сколько времени?» можно считать нейтральным, а «Скажи уже, наконец, сколько времени?» негативным.

Для того чтобы научить систему отвечать по новым классам сообщений, необходимо обучить классификатор *wit* на этих классах: правилам, по которым система должна отвечать на сообщение, и фразам, с помощью которых выводится ответное сообщение по правилам. В данной лабораторной работе необходимо обучить классификатор для ответа на сообщения. Система научится отвечать на новые сообщения после создания логических правил и фраз, т. е. после выполнения лабораторной работы № 3.

Ход работы

1 После выбора темы предметной области и написания ее формальной спецификации можно приступить к подбору и разработке различных вариантов диалогов. В данной лабораторной работе достаточно привести диалог с системой, состоящий из восьми сообщений. Например, диалог с системой NIKA, состоящий из четырех сообщений, может выглядеть таким образом, как это показано на рисунке 14.

2 На рисунке видно, что Сидоров Иван сначала приветствует систему NIKA. Система NIKA отвечает взаимностью. После собеседник Иван узнает

назначение системы. Далее можно приступить к обучению системы и описанию ответных фраз.

3 Классификатор сообщений по теме системы NIKA использует открытый облачный сервис Wit.ai [11], который позволяет получать класс намерения сообщения (*intent*), сущности сообщения (*entity*) и класс признака (*trait*). Наглядная инструкция по использованию Wit.ai есть на сайте, а также на других многочисленных интернет-ресурсах. Далее будет описано, как пользоваться Wit.ai, а также подключить свое IT-приложение к системе.

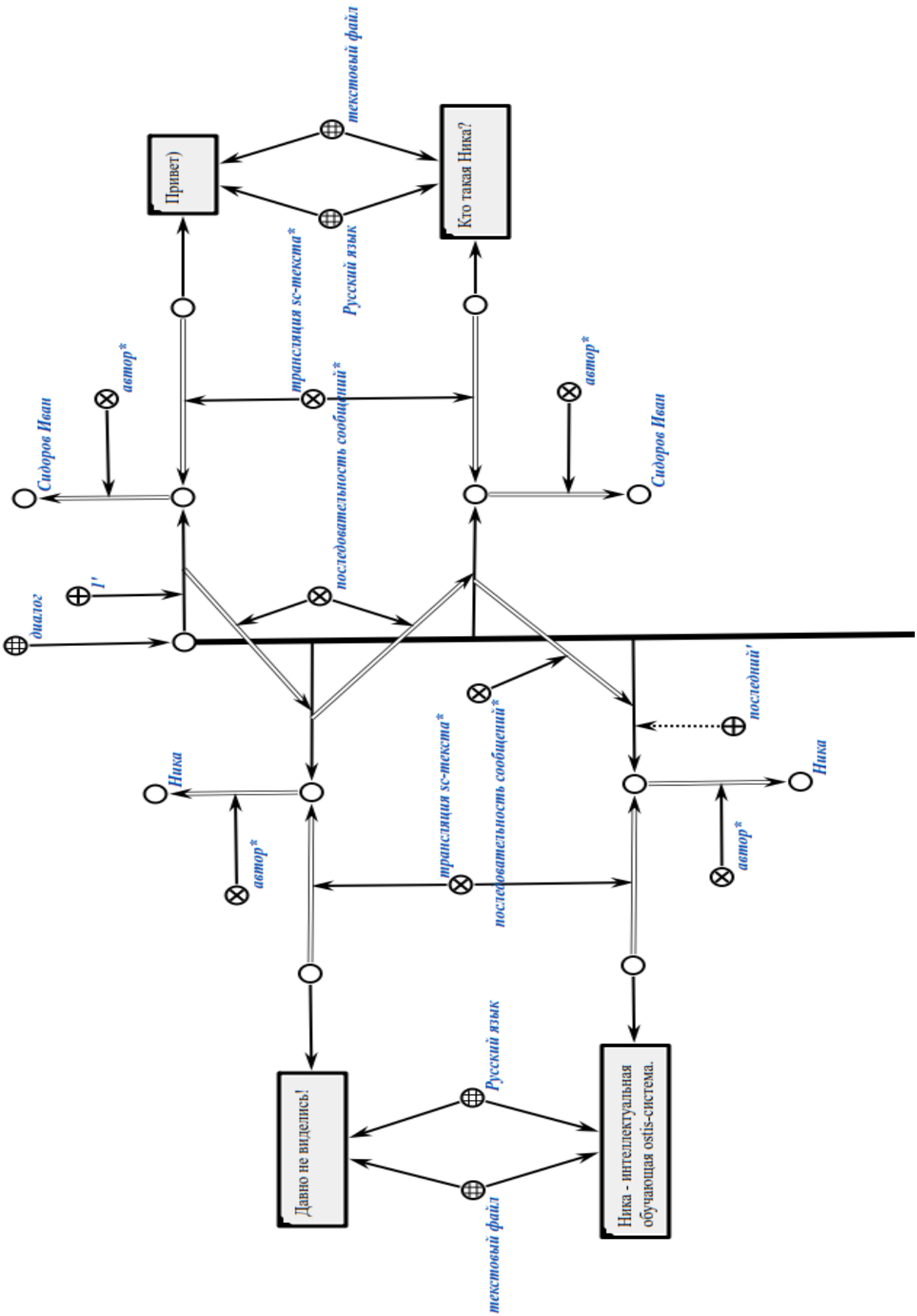

Рисунок 14 – Пример диалога с системой NIKA на SCg

Создание собственного приложения классификации Wit.ai

Переходим на Wit.ai [11], входим с помощью Facebook (рисунок 15). Если через Google Chrome не удается войти, используйте другой веб-браузер (например, Mozilla Firefox).

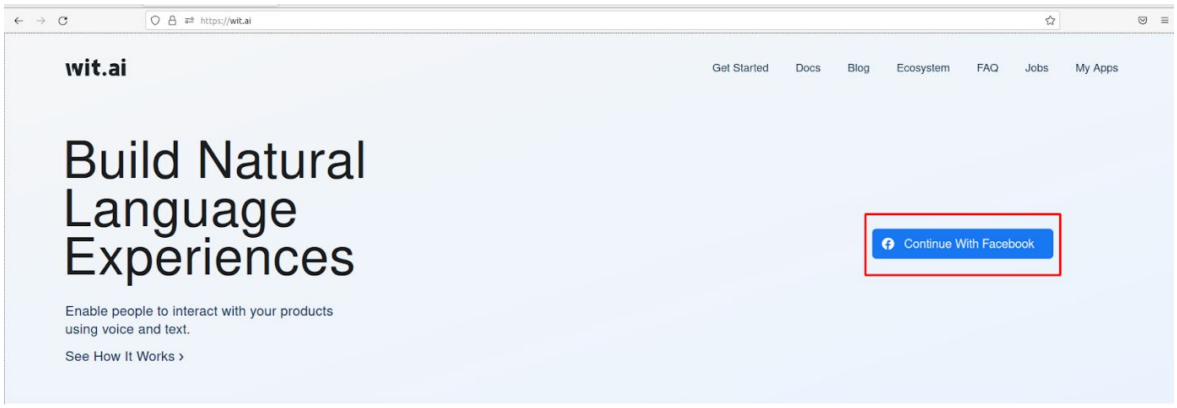

Рисунок 15 – Главное окно Wit.ai

Создание нового приложения представлено на рисунке 16.

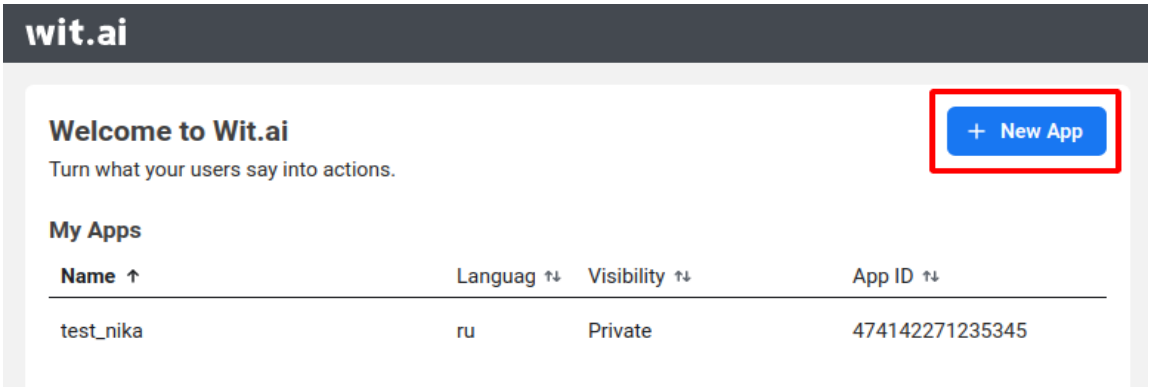

Рисунок 16 – Создание приложения для классификации

Заполняем поле с именем приложения, вписав туда, например, название предметной области, выбираем язык и приватность приложения (рисунок 17).

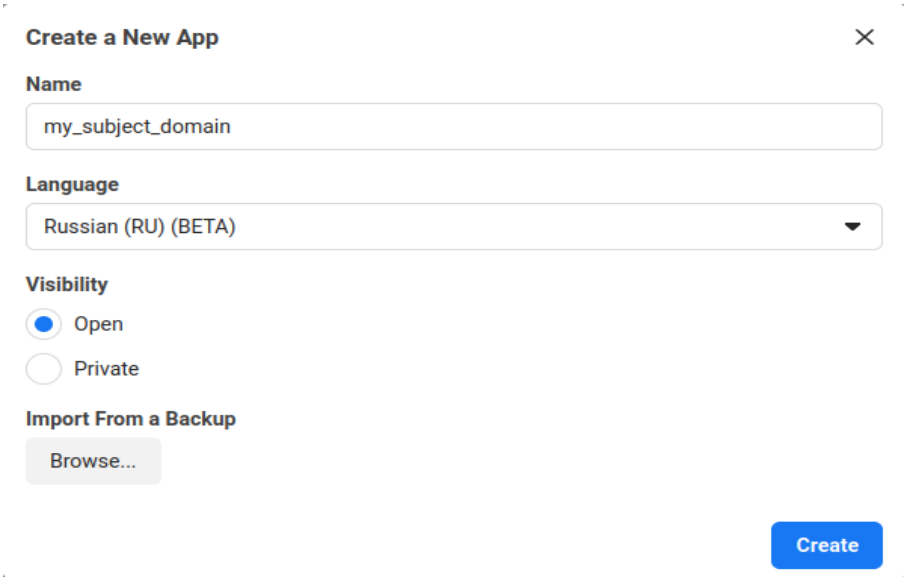

Рисунок 17 – Окно настройки создаваемого приложения

Обучение классификатора

Затем во вкладке *Management/Intents* добавляем классы сообщений, которые мы хотим распознавать (рисунок 18).

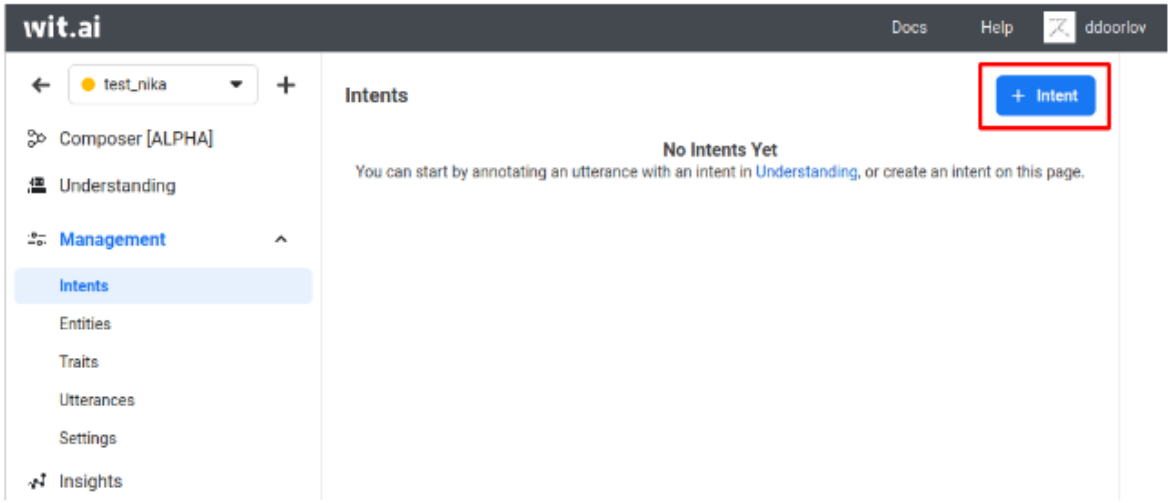

Рисунок 18 – Создание классов намерений сообщений (wit traits)

Во вкладке *Entities* необходимо создать роли сущностей, которые мы хотим выделять в сообщении (рисунок 19). При работе агента классификации

используется только одна роль *rrel_entity*, остальная информация используется из базы знаний.

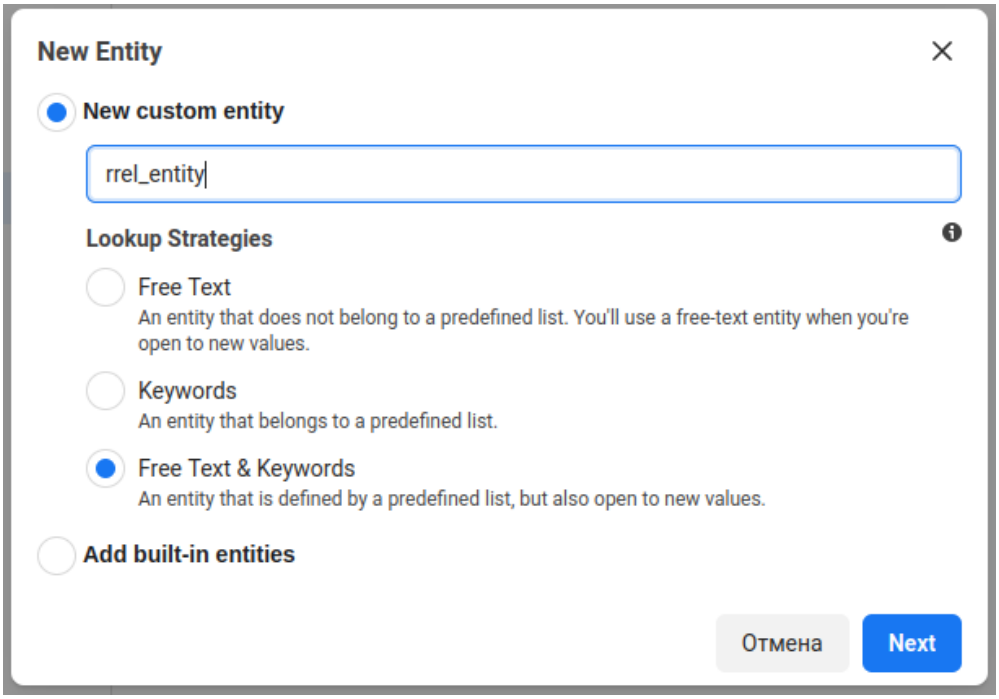

Рисунок 19 – Создание ролей сущностей (wit entities)

Переходим в *Understanding*, вводим разные по формулировке сообщения, которые хотим классифицировать и выбираем их классы, а также выделяем сущности (рисунок 20).

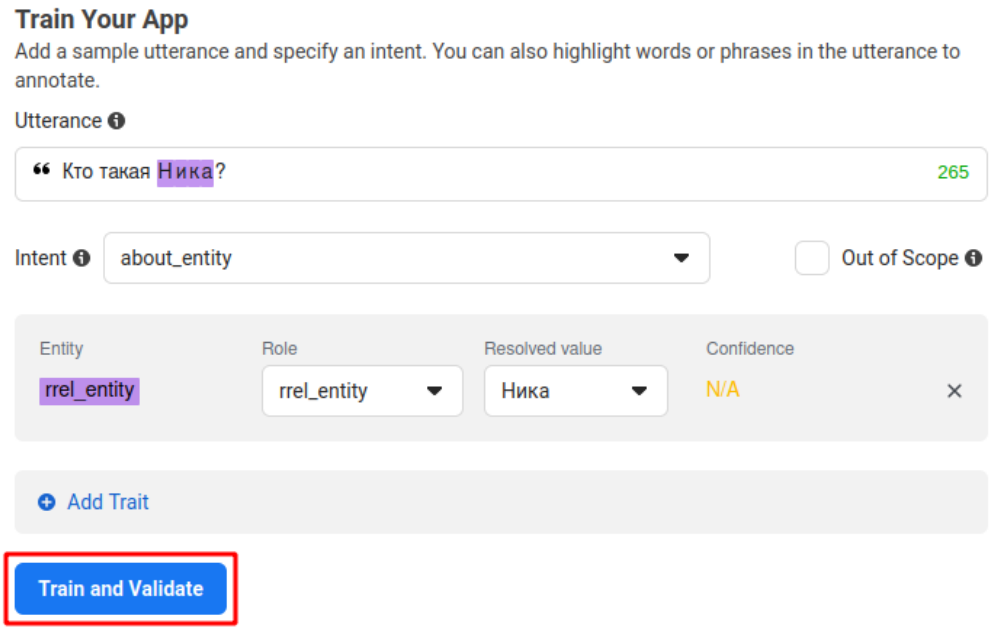

Рисунок 20 – Обучение приложения Wit

На данном примере можно использовать такие формулировки как «Что ты знаешь о NIKA?», «Что ты можешь сказать о NIKA?» и другие.

При обучении можно выделять не только одну сущность, но также и несколько сущностей сразу. В данной ситуации необходимо выделить подходящие роли (*intents*) для выделяемых сущностей (чтобы отличать их между собой, если это необходимо). Если вы создаете свою новую роль, то ее необходимо формализовать в базе знаний. Например, в формулировке вопроса: «Какая погода в Бресте в Беларуси?» (рисунок 21). Для ответа необходимо выделить город Брест и страну Беларусь. Заметьте, что выделяемые сущности лучше распознавать как существительные в именительном падеже.

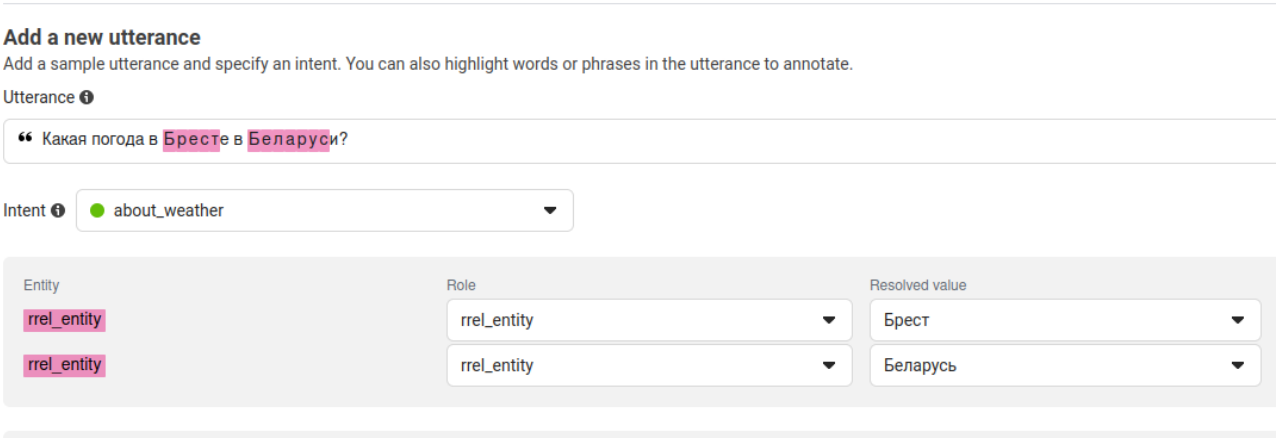

Рисунок 21 – Пример формализации вопроса в базе знаний

После этого приложение способно классифицировать похожие сообщения и выделять его сущности. Проверить это можно, вводя новые сообщения в окно *Utterance*, классы и сущности должны определяться автоматически.

Обучение может занять некоторое время, прогресс обучения можно наблюдать в верхнем левом углу на странице приложения Wit.ai.

Подключение классификатора к системе NIKA

Переходим в настройки и копируем *Server Access Token* (рисунок 22).

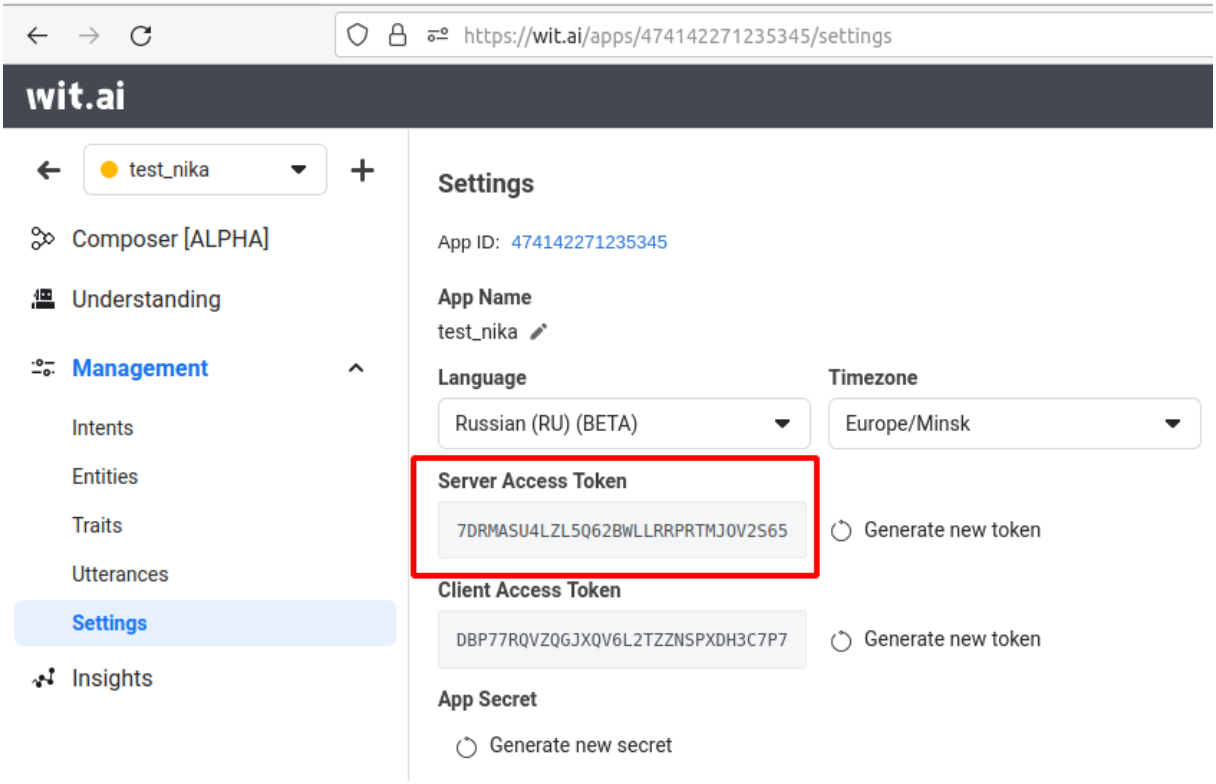

Рисунок 22 – Получение Server Access Token

Затем записываем данный токен в конфигурацию по пути *nika/nika.ini* в переменную *server_token*:

```
[wit-ai]
server_token = 7DRMASU4LZL5Q62BWLLRRPRTMJOV2S65
url = https://api.wit.ai/message
```

Далее нужно формализовать новое классифицируемое сообщение. Здесь важно, чтобы содержимое ссылки под отношением *nrel_wit_ai_idtf* было в точности таким же, как и в приложении Wit.ai. Фрагмент формализации приведен ниже:

```
concept_message_about_entity
<- sc_node_not_relation;
=> nrel_main_idtf:
   [сообщение о сущности]
   (* <- lang_ru;; *);
=> nrel_wit_ai_idtf:
   [about_entity];
<- rrel_key_sc_element:
   ...
  (*
     => nrel_main_idtf:
        [Опр. (сообщение о сущности)]
```

```
        (* <- lang_ru;; *);;
     <- definition;;
     <= nrel_sc_text_translation:
     ...
     (*
        -> rrel_example:
        [Сообщение о сущности - атомарное сообщение, которое
        содержит сушность, информацию о которой необходимо найти.]
        (* <- lang_ru;; *);;
     *);;
  <= nrel_using_constants:
     {
       concept_atomic_message
     };;
  *);;
```

Далее во множество распознаваемых сообщений (*concept_intent_possible_class*) необходимо добавить новое классифицируемое сообщение:

```
concept_intent_possible_class
<- sc_node_class;
=> nrel_main_idtf:
   [возможный класс цели Wit.ai]
   (* <- lang_ru;; *);
<- rrel_key_sc_element:
    ...
    (*
       => nrel_main_idtf:
          [Опр. (возможный класс цели Wit.ai)]
           (* <- lang_ru;; *);;
     <- definition;;
     <= nrel_sc_text_translation:
        ...
        (*
           -> rrel_example:
              [Возможный класс цели Wit.ai - класс, элементы
              которого могут быть выделены посредством Wit.ai как
              цели.]
```

```
                (* <- lang_ru;; *);;
          *);;
        *);
 -> concept_greeting_message;
 -> concept_message_about_entity;;
```

Аналогичным образом формализуются классы признаков (*traits*). Необходимо указать идентификатор *wit* и добавить класс во множество *concept_trait_possible_class*. Классификация при использовании *traits* является необязательной, а при использовании *intent* – обязательной.

Для выделения сущностей сообщений необходимо формализовать класс выделяемых сущностей и их роли в сообщении (рисунок 23).

Рекомендуется использовать графический редактор баз знаний – редактор KBE [17]. Сохраненные в формате *.gwf* файлы также можно погружать в базу знаний.

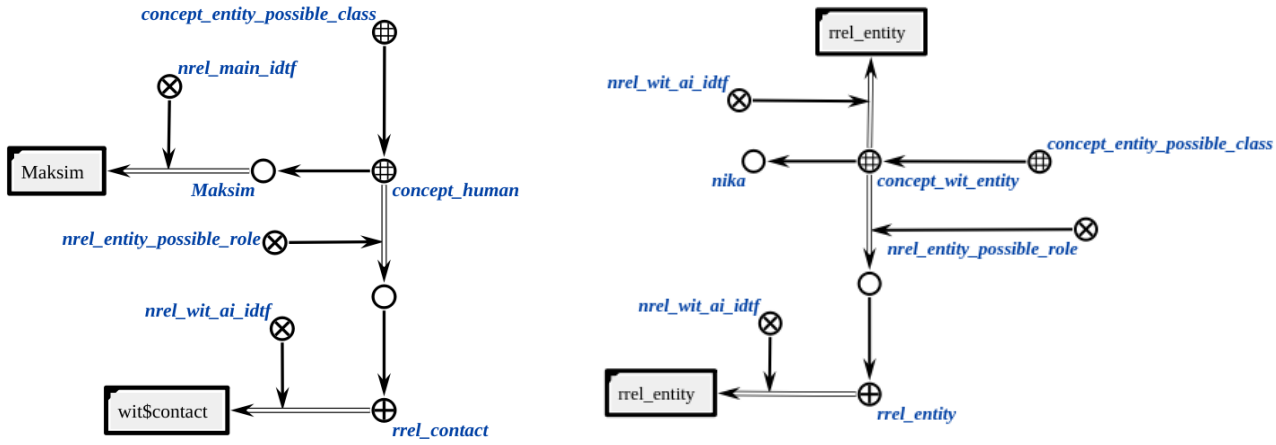

Рисунок 23 – Пример формализации конструкции, необходимой для выделения сущности

Сервис Wit.ai предоставляет некоторые встроенные сущности и классы, например выделение в сообщении персоны, времени, дистанции, числа, места, суммы денег, электронной почты, признака сообщения по включению/выключению чего-либо и другие (рисунок 24). Пользоваться встроенными сущностями и классами разрешается, однако не стоит этим злоупотреблять (максимум две сущности).

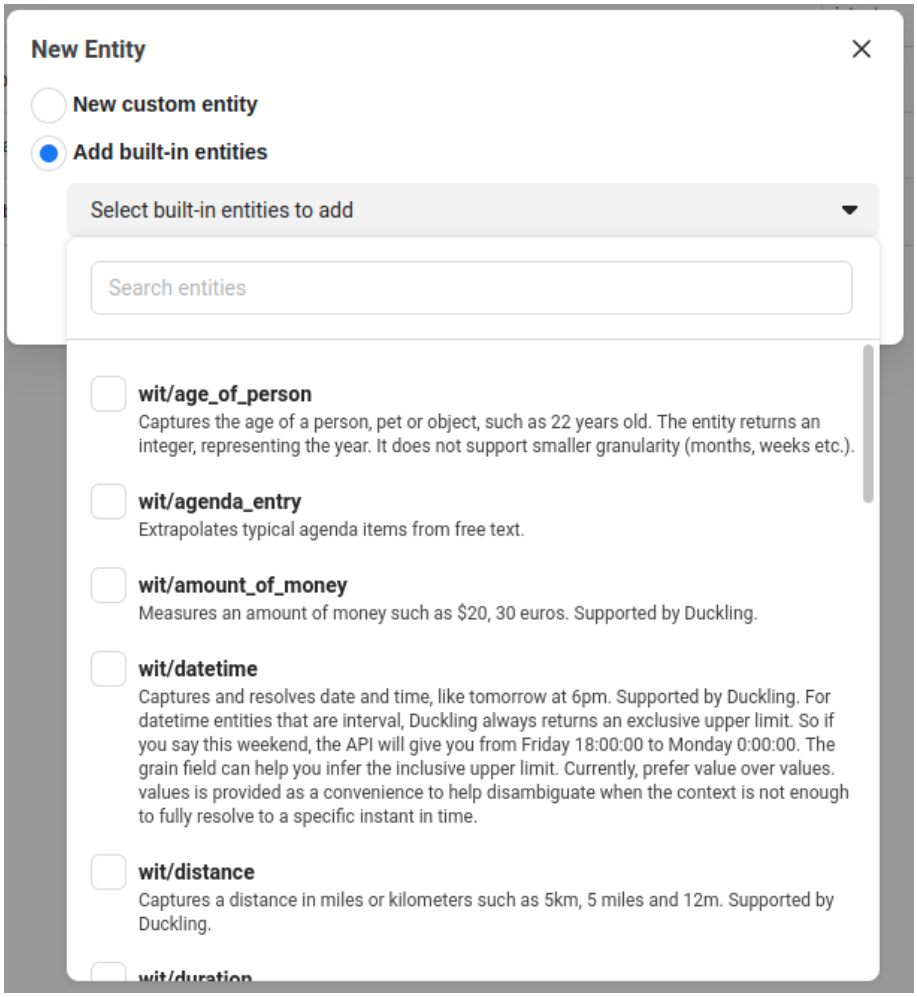

Рисунок 24 – Заранее обученные сущности Wit

Для более полного понимания механизма классификации рекомендуется изучить документацию агента классификации NIKA [18], исходный код агента классификации [19] и связанные с ним фрагменты базы знаний.

После выполнения лабораторной работы, необходимо в папку *kb/extra/{папка с исходниками предметной области}/classification/* загрузить исходники с результатами классификации сообщений, предварительно сделав экспорт данных из Wit.ai в формате *json* (рисунок 25).

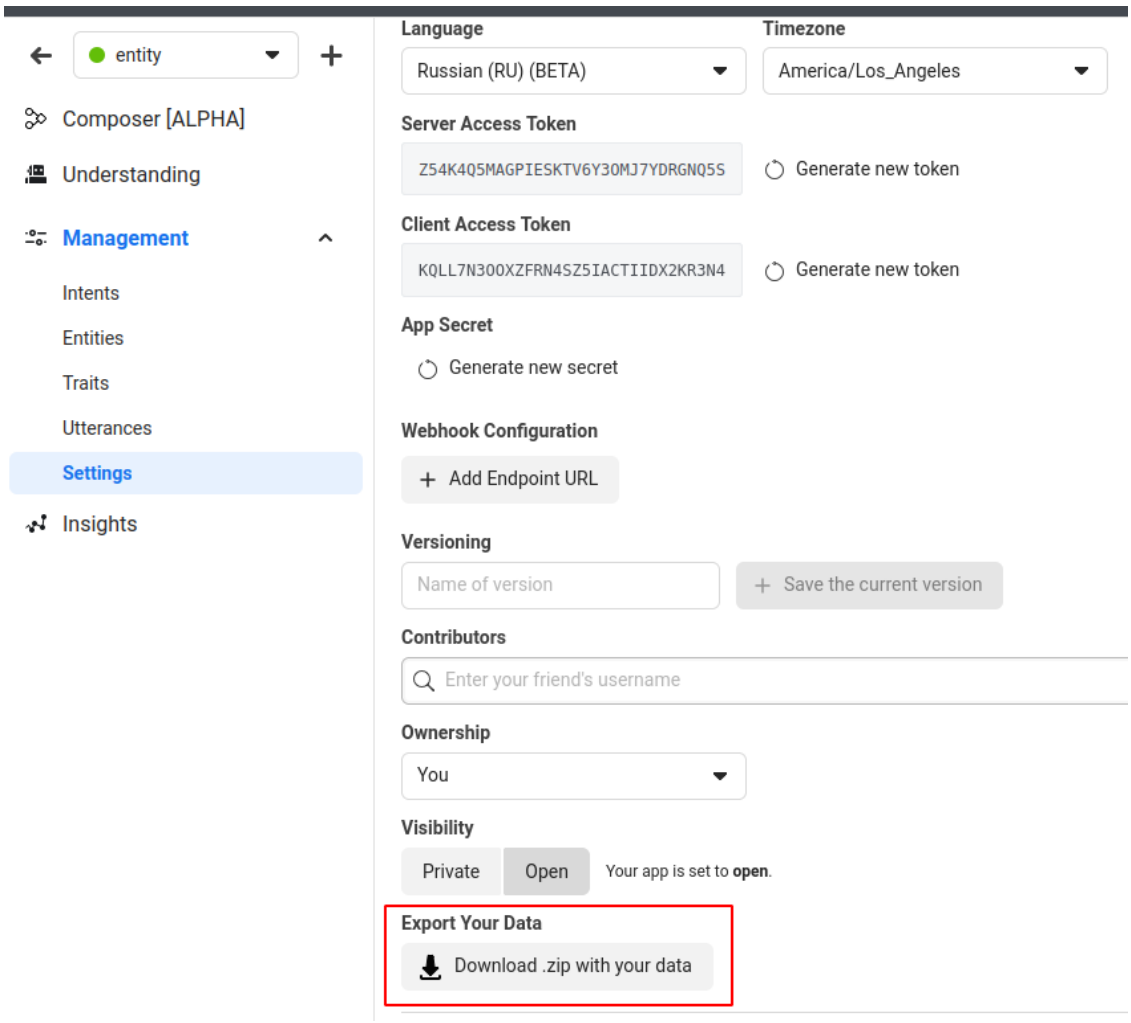

Рисунок 25 – Экспорт данных Wit.ai

Подготовка к защите

1 Протестировать классификацию сообщений и выделение сущностей, используя Wit.ai utterances.

2 Протестировать сообщения с помощью вопроса: «Что такое …?» для каждой сущности.

3 Сделать отчет (электронная версия) с иллюстрацией тестирования.

4 Загрузить изменения в свой форк на GitHub со всеми выполненными заданиями.

5 Подготовиться к ответу на вопросы и решению задач. Повторить теоретический материал.

Ход защиты

1 Проверка отчета.

2 Иллюстрация и тестирование классификатора Wit.ai.

3 Ответы на вопросы преподавателя и решение задач.

4 Выставление оценки.

Вопросы для подготовки к защите

1 Что такое классификация? Какая цель задачи классификации? Как можно классифицировать сообщение?

2 Какие функциональные возможности у сервиса Wit.ai? Какой принцип работы Wit.ai?

3 Алфавит ядра SC-кода и алфавит SC-кода: что из себя представляют, в чем различие?

4 На чем основаны направления расширения ядра SC-кода?

5 Приведите классификацию интеллектуальных систем.

6 Что такое экспертная система? В чем ее отличие от других систем?

7 Какие существуют формы представления знаний? Какая форма представления знаний используется в ходе выполнения лабораторной работы?

8 Какие требования нужно учитывать при составлении и формализации диалога с компьютерной системой?

Задача

Формализовать сообщение с выделением сущности:

(1) Поставь будильник на 7 утра.

(2) Сколько стоит арбуз на рынке?

(3) Сколько ядер у процессора Intel Core i9-12900K?

(4) Что ты знаешь о 9 «В» классе?

(5) Расскажи о треугольнике ABC.

(6) Алиса, включи, пожалуйста, свет на кухне.

Пример решения задачи (6) приведен на рисунке 26.

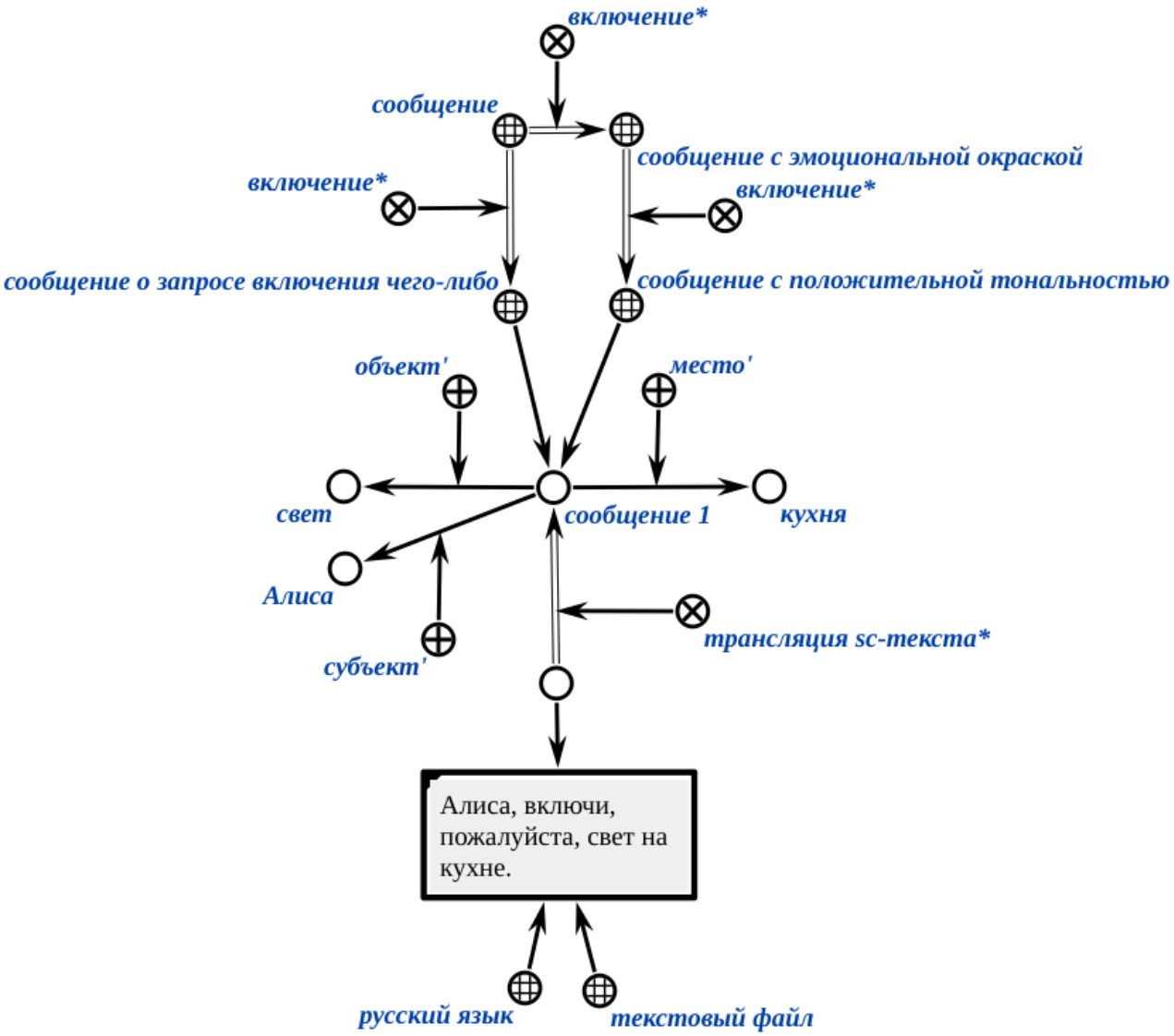

Рисунок 26 – Пример формализованного сообщения «Алиса, включи, пожалуйста, свет на кухне»

Лабораторная работа № 3. Проектирование логических правил для ответа на сообщения системы NIKA

Цель: изучить принципы работы механизма ответа на сообщения, используя логические правила и фразы, приобрести навыки проектирования логических формул для ведения диалога.

Задание

1 Спроектировать логические правила для ответа на сообщения, покрывая каждый класс фраз для ответа (см. ход работы на с. 105):

– правила должны использовать не только классы сообщений, но и выделяемые в сообщении сущности из первой лабораторной работы;

− необходимо продемонстрировать различные ответы на одно и то же сообщение в зависимости от состояния базы знаний, для этого необходимо использовать фразы с шаблонами из второй лабораторной работы.

− обязательным является условие получения ответа на сообщение, которое можно увидеть, используя пользовательский интерфейс системы NIKA.

2 Описать фразы ответных сообщений в базе знаний:

− фразы сообщений не должны повторяться;

− для каждого класса ответных фраз должно быть не менее двух фраз и хотя бы три фразы на весь диалог с использованием шаблона ответа.

3 По результатам работы составить отчет, где указать цель, задание, вариант и показать тестирование сценария диалога.

4 Ответить на вопросы и решить задачи, указанные в конце лабораторной работы.

5 Загрузить изменения в свой форк.

Теоретические сведения

Экспертные системы моделируют работу эксперта, т. е. человека, обладающего большим опытом в некоторой предметной области. Экспертные системы позволяют заменить дорогостоящего эксперта путем переноса его знаний в базу знаний экспертной системы. Наиболее ценными знаниями в такой ситуации являются высказывания (как фактографические, так и логические формулы). Используя высказывания, можно получать новые знания (новые формулы) на основе существующих с помощью аппарата логического вывода. Например, есть общая схема рассуждения, правило вывода **Modus ponens**, которое формально можно сформулировать следующим образом: если есть истинное знание A, а также знание, что из A следует B (A влечет B, A выводит B), то истинным является и знание B (рисунок 27). Таким образом, при наличии в базе знаний структур, соответствующих данному правилу вывода, можно получать новые знания в рамках некоторой формальной теории. Например, эксперт сформулировал такое правило из предметной области геометрии Евклида: если треугольник имеет угол, градусная мера которого равна 90°, такой треугольник является прямоугольным. Другим примером может быть теорема Пифагора, теорема синусов, формула расчета площади прямоугольного

треугольника и другие. Используя такие правила и знания о предметных объектах, можно получить новые знания, изначально неочевидные.

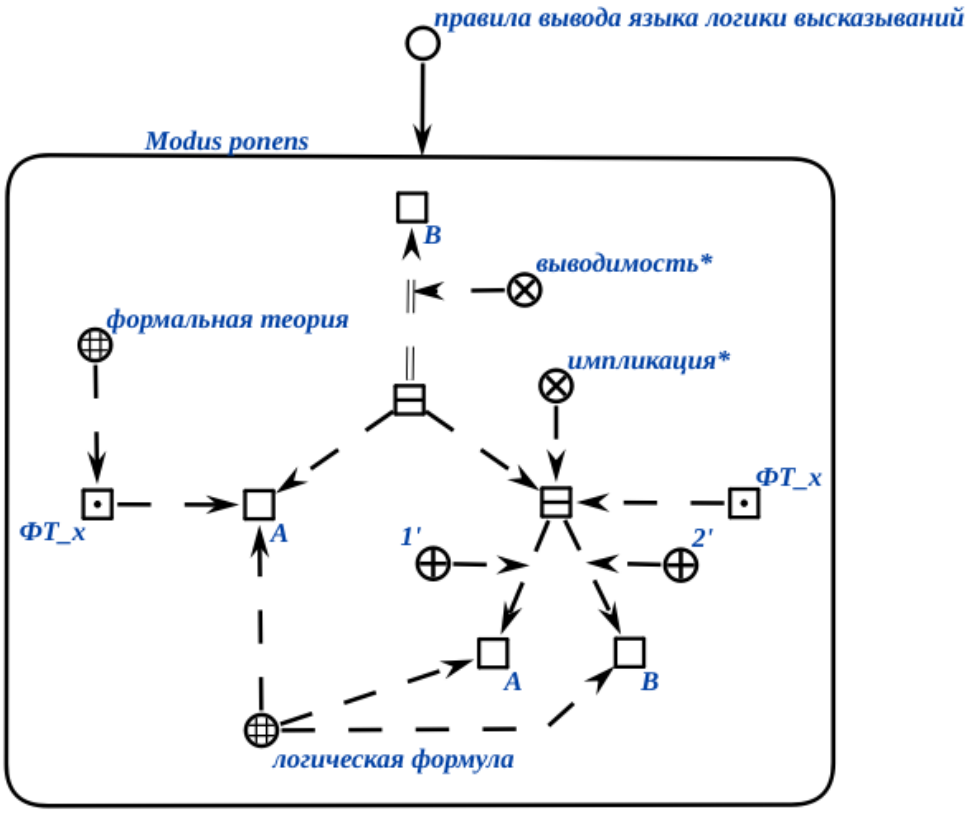

Рисунок 27 – Пример формализации правила вывода Modus ponens

Логическая связка – это отношение, связками которого являются высказывания.

Конъюнкция – это множество конъюнктивных высказываний, каждое из которых истинно в рамках некоторой формальной теории только в том случае, когда все его компоненты истинны в рамках этой же формальной теории.

Дизъюнкция – это множество дизъюнктивных высказываний, каждое из которых истинно в рамках некоторой формальной теории только в том случае, когда хотя бы один его компонентов является истинным в рамках этой же формальной теории.

Отрицание – это множество высказываний об отрицании, каждое из которых истинно в рамках некоторой формальной теории только в том случае, когда его единственный элемент является ложным в рамках этой же формальной теории.

Импликация – это множество импликативных неатомарных высказываний, каждое из которых состоит из посылки (первый компонент высказывания) и следствия (второй компонент высказывания). На рисунке 28 показан пример импликации, которая содержит следующую информацию: для любых переменных ***_triangle*** и ***_angle*** если ***_triangle*** является прямоугольным треугольником, то синус его внутреннего угла *_angle* равен единице.

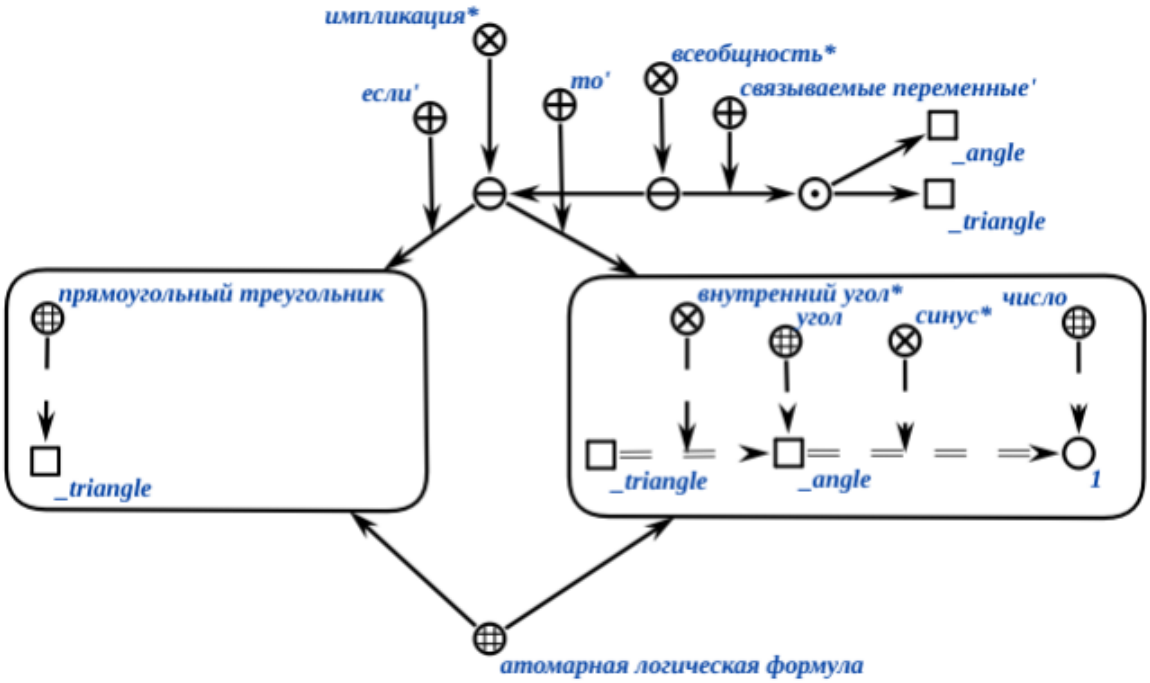

Рисунок 28 – Пример импликации

Другим простейшим правилом вывода, используемым в экспертных системах, является *правило резолюции*. Оно гласит, что если истинно A или B, а также истинно C или не B, то истинно A или C (рисунок 29).

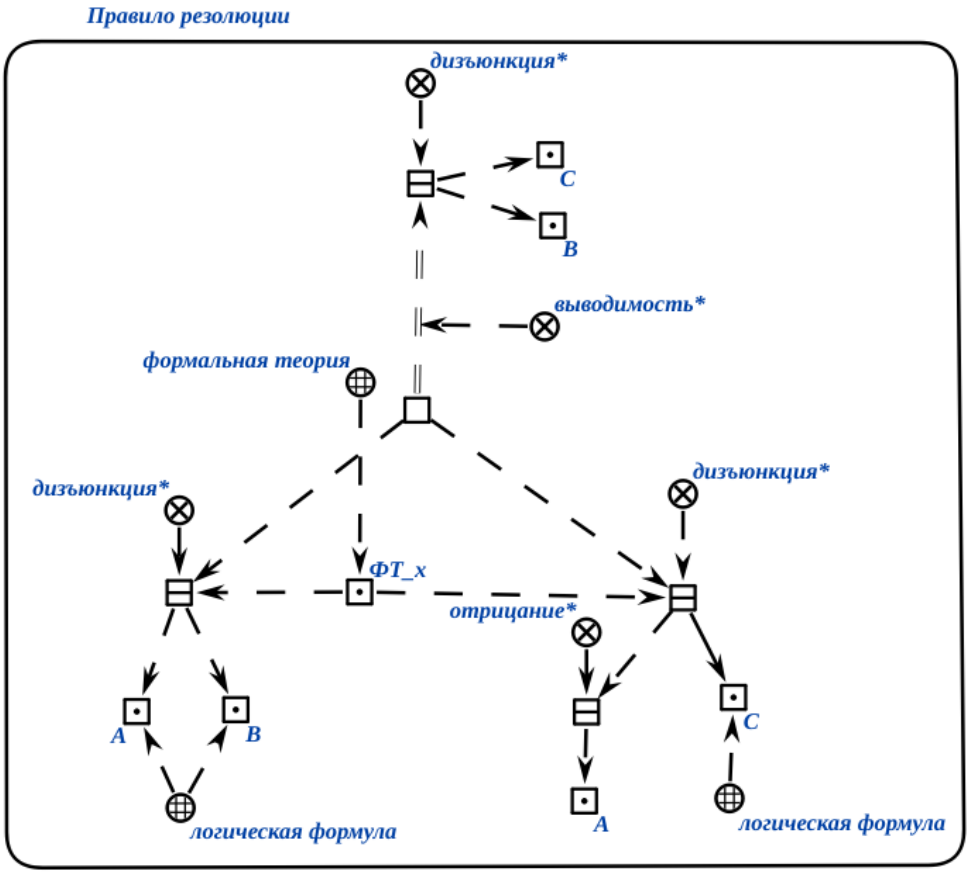

Рисунок 29 – Пример формализации правила резолюций

Формальная теория – это множество высказываний, которые считаются истинными в рамках данной формальной теории. Некоторые высказывания считаются аксиомами, а другие доказываются на основе других высказываний в рамках этой же формальной теории.

Под ***высказыванием*** понимается некоторая структура (в которую входят sc-константы из некоторой предметной области и/или sc-переменные) или логическая связка, которая может трактоваться как истинная или ложная в рамках какой-либо предметной области:

высказывание
=> *разбиение*:*
{
- атомарное высказывание
- неатомарное высказывание

}
=> *разбиение*:*
{
- фактографическое высказывание

- логическая формула

}

Атомарное высказывание – это высказывание, которое содержит хотя бы один sc-элемент, не являющийся знаком другого высказывания.

Неатомарное высказывание – это высказывание, в состав которого входят только знаки других высказываний. Нельзя говорить об истинности либо ложности неатомарного высказывания в рамках какой-либо формальной теории в случае, когда невозможно установить истинность либо ложность любого из его элементов в рамках этой же формальной теории. Высказывание также может быть бессмысленным в случае, если в какое-либо атомарное высказывание входит какая-либо sc-константа, не являющаяся элементом предметной области, описываемой формальной теории.

Под ***фактографическим высказыванием*** понимается атомарное высказывание, в состав которого не входит ни одна sc-переменная или ни одно атомарное высказывание, все элементы которого также являются фактографическими высказываниями.

логическая формула

=> разбиение:*

{

- атомарная логическая формула
- неатомарная логическая формула

}

Под логической формулой понимается:

− атомарное высказывание, в состав которого входит хотя бы одна sc-переменная;

− неатомарное высказывание, хотя бы один элемент которого является логической формулой.

Под атомарной логической формулой понимается атомарное высказывание, которое является логической формулой. Атомарная логическая формула не содержит логических связок.

По умолчанию атомарная логическая формула трактуется как высказывание о существовании, т. е. наличии в памяти значений, соответствующих всем sc-переменным, входящим в состав данной формулы, и не попадающих под действие какого-либо другого квантора (указанного явно

или по умолчанию). Таким образом, на все sc-переменные, входящие в состав атомарной логической формулы и не попадающие под действие какого-либо другого квантора, неявно накладывается квантор существования. По умолчанию на все переменные, входящие в обе части высказывания об импликации (или хотя бы одну из *подформул** каждой части), неявно накладывается квантор *всеобщности**, при условии что эти переменные не связаны другим квантором, указанным явно. То же самое справедливо для эквиваленции.

Для описания сложных закономерностей в базе знаний используют неатомарные логические формулы, представляющие собой комбинацию различных логических связок и кванторов. На рисунке 30 представлен пример формулы, которая иллюстрирует теорему о сумме геометрических отрезков.

Фразы для ответа

Фраза для ответа на сообщение представляет собой sc-ссылку, содержимое которой используется при генерации ответного сообщения на сообщение пользователя. Для фразы обязательно указывается язык путем добавления фразы во множество соответствующего языка (проведением дуги принадлежности от класса языка, например *lang_ru*, к фразе). Также для фразы указывается как минимум один класс, который описывает тему фразы (например, приветствие и позитивное приветствие). Фразы могут быть простыми (рисунок 31), без использования шаблона, и сложные (рисунок 32), с использованием шаблона. Шаблоны позволяют учитывать состояние базы знаний при ответе на сообщение. Чем больше переменных во фразе, тем более вариативный и осмысленный получается ответ.

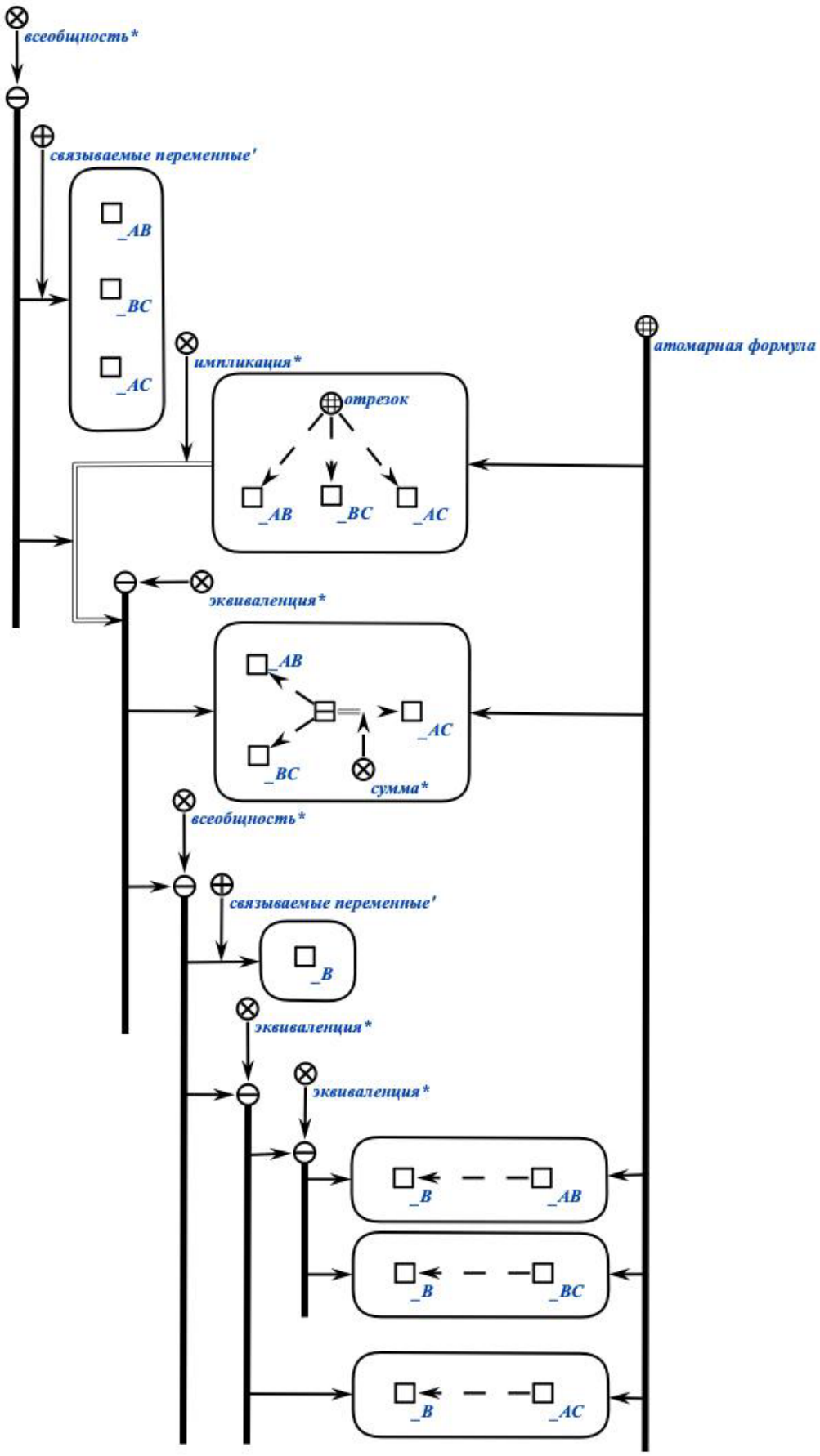

Рисунок 30 – Пример логической формулы о сумме геометрических отрезков

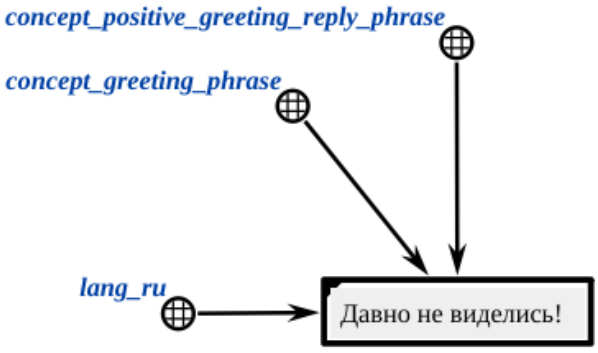

Рисунок 31 – Формализация фразы приветствия без использования шаблона

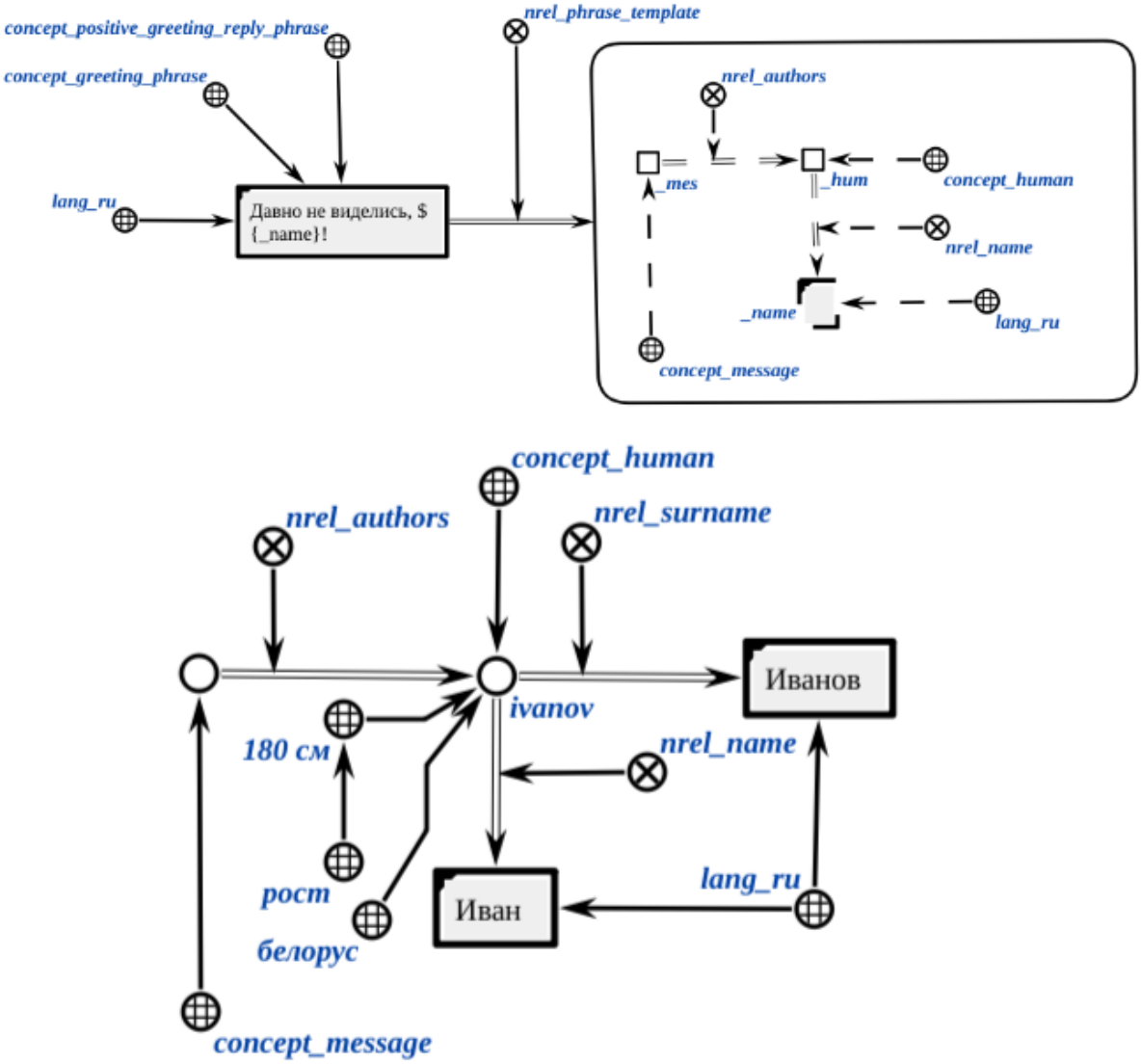

Рисунок 32 – Формализация фразы приветствия с использованием шаблона и соответствующий фрагмент базы знаний

Для того чтобы подставить в sc-ссылку ответной фразы содержимое ссылки из базы знаний, нужно использовать фигурные скобки, в которые заключен идентификатор той ссылки, содержимое которой подставляется в

ответную фразу. На рисунке 33 такой ссылкой является ссылка с идентификатором *_name*.

Для того чтобы система при ответе на сообщение могла воспользоваться фразой с шаблоном, необходимо, чтобы в базе знаний хранились конструкции, которые находятся в шаблоне, иначе система не сможет ответить по желаемому правилу и ответом станет фраза непонимания («Я не понимаю Вас»).

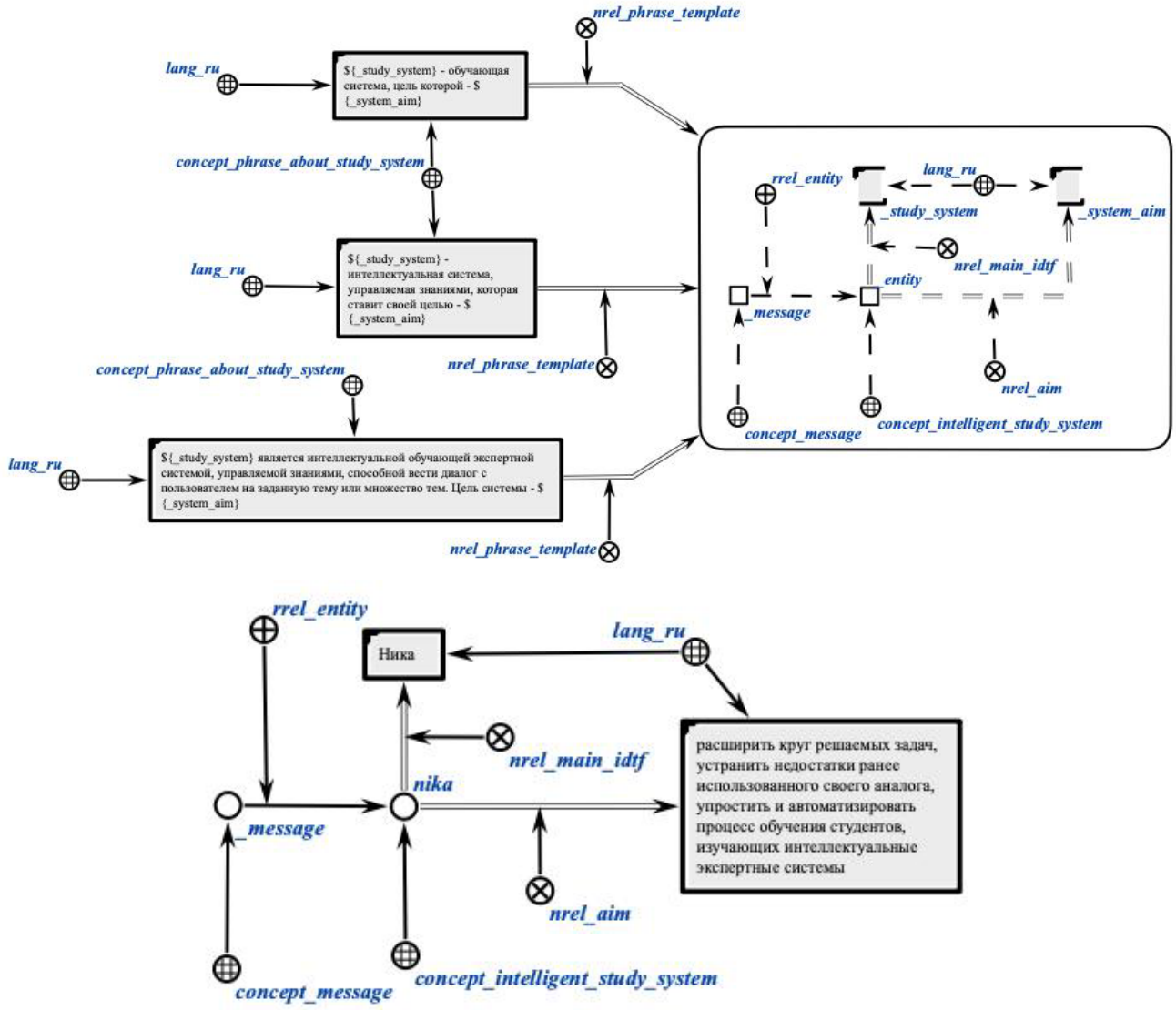

Рисунок 33 – Формализация фразы про обучающую систему с использованием шаблона

Ход работы

Формализация логических правил. После выполнения первой и второй лабораторных работ база знаний экспертной обучающей системы NIKA содержит формализованную предметную область, включает множество сообщений, с помощью которых система сможет отвечать на сообщения

пользователя. В данной лабораторной работе предлагается спроектировать логические правила и фразы для ответа, с помощью которых система сможет использовать полученные фразы для ведения диалога. Пример простейшего правила приведен на рисунке 34.

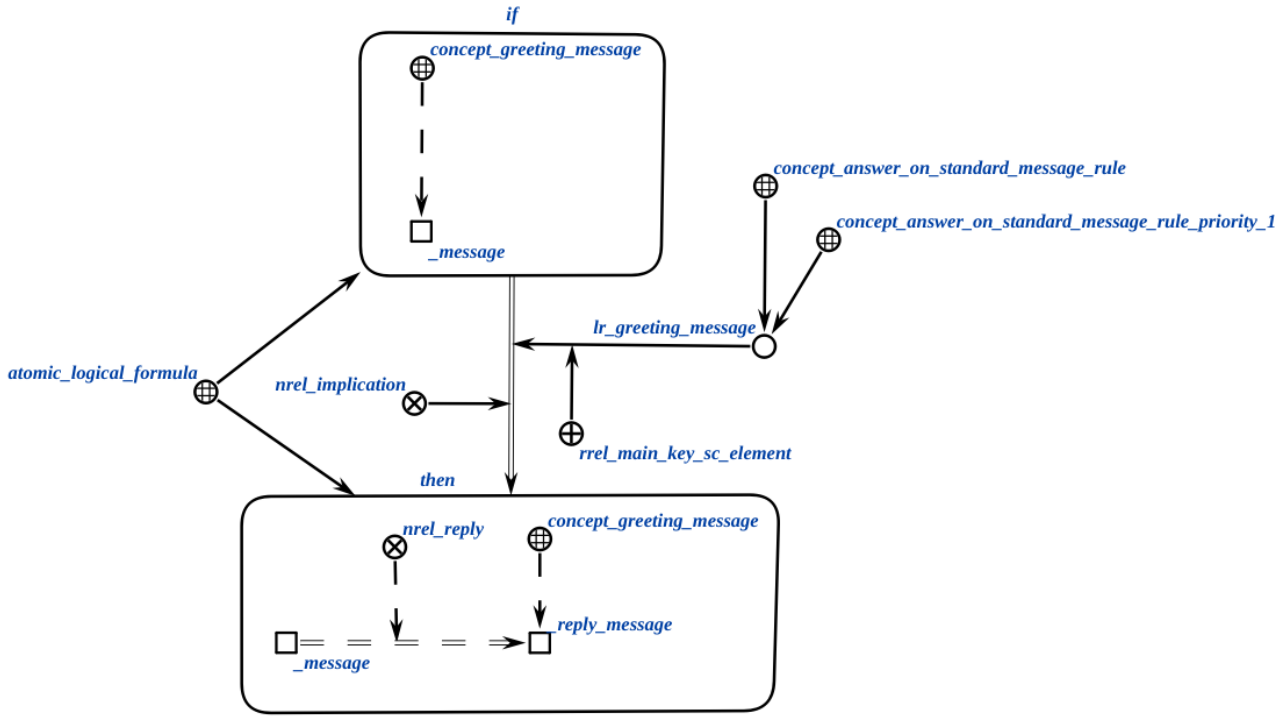

Рисунок 34 – Пример логического правила для ответа на приветственное сообщение

Кроме того, можно использовать логическое правило для ответа на сообщение про условие или дедлайн неизвестной лабораторной работы с использованием логической связки ИЛИ, выраженной с помощью неролевого отношения *nrel_disjunction* (рисунок 35).

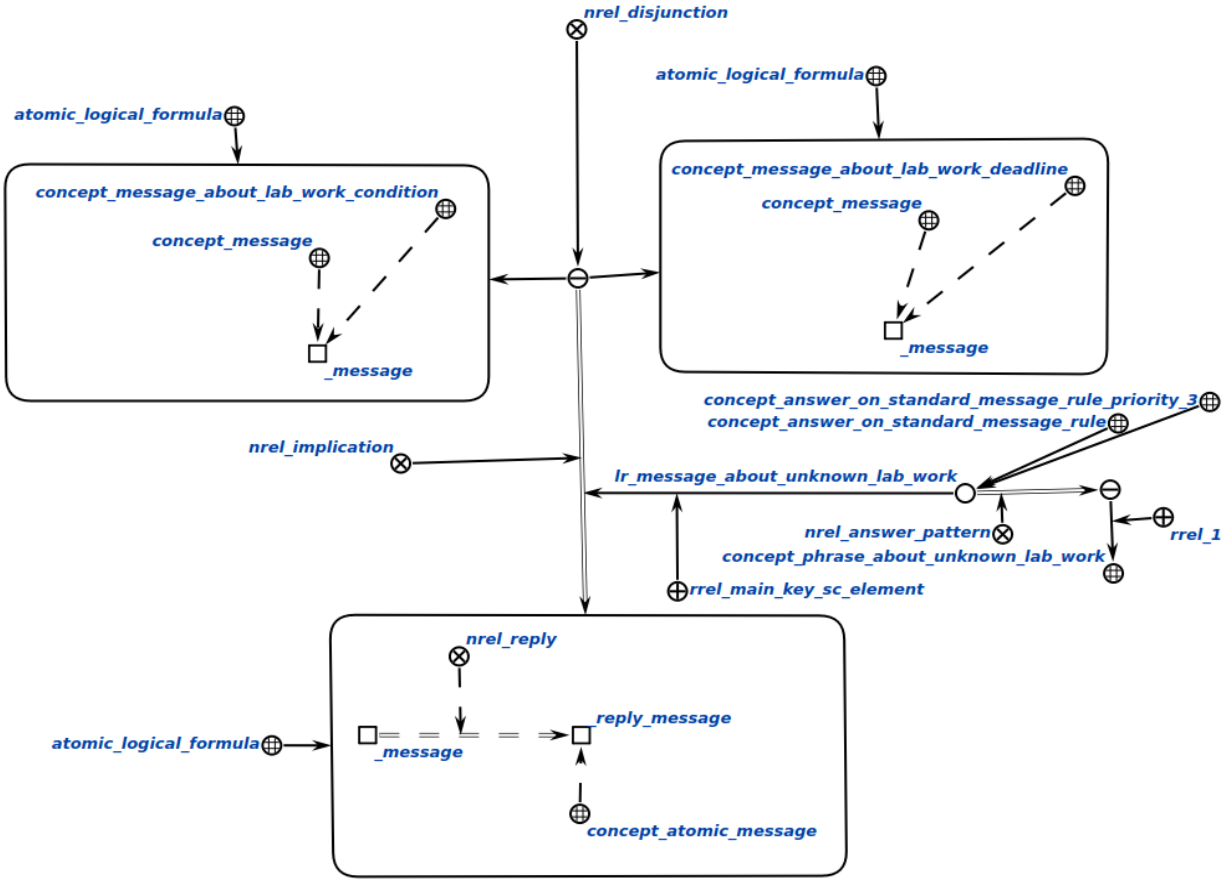

Рисунок 35 – Пример логического правила с одной логической связкой импликации

Логическое правило на рисунке 35 состоит из одной логической связки импликации (правило вида «***Если …, то…***»). Посылкой («если») данной импликации является атомарная логическая формула (sc-контур с переменными), содержащая переменную *_message* и один класс, которому она принадлежит: *concept_greeting_message*. Следствием («то») данной формулы является контур, содержащий информацию о том, что ответом (*nrel_reply*) на такое сообщение должно быть сообщение класса *concept_greeting_message*. Таким образом, при появлении в памяти конструкции из посылки (т. е. при отправке сообщения, которое было классифицировано как сообщение приветствия) генерируется конструкция из следствия (появляется новый узел сообщения, являющийся ответом на поступившее сообщение пользователя и принадлежащее классу приветственного сообщения). Как посылка, так и следствие могут быть насколько угодно большими и сложными, однако для того, чтобы правило сработало, обязательно в памяти системы должна быть конструкция из посылки импликации.

Каждое логическое правило должно иметь ***шаблон ответа*** (*nrel_answer_pattern*) – это множество, которому принадлежит тот класс фраз, который используется при ответе на сообщение по данному правилу (рисунок 36).

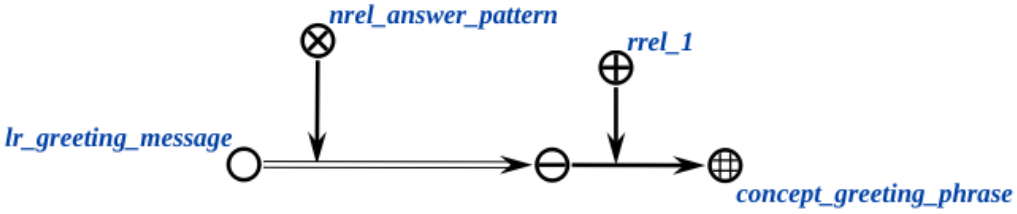

Рисунок 36 – Шаблон ответа на сообщение в соответствии с правилом

Таким образом, используя правило на рисунке 35, шаблон фразы на рисунке 34 и фразы на рисунке 37, можно получить ответ от системы, как на рисунке 38.

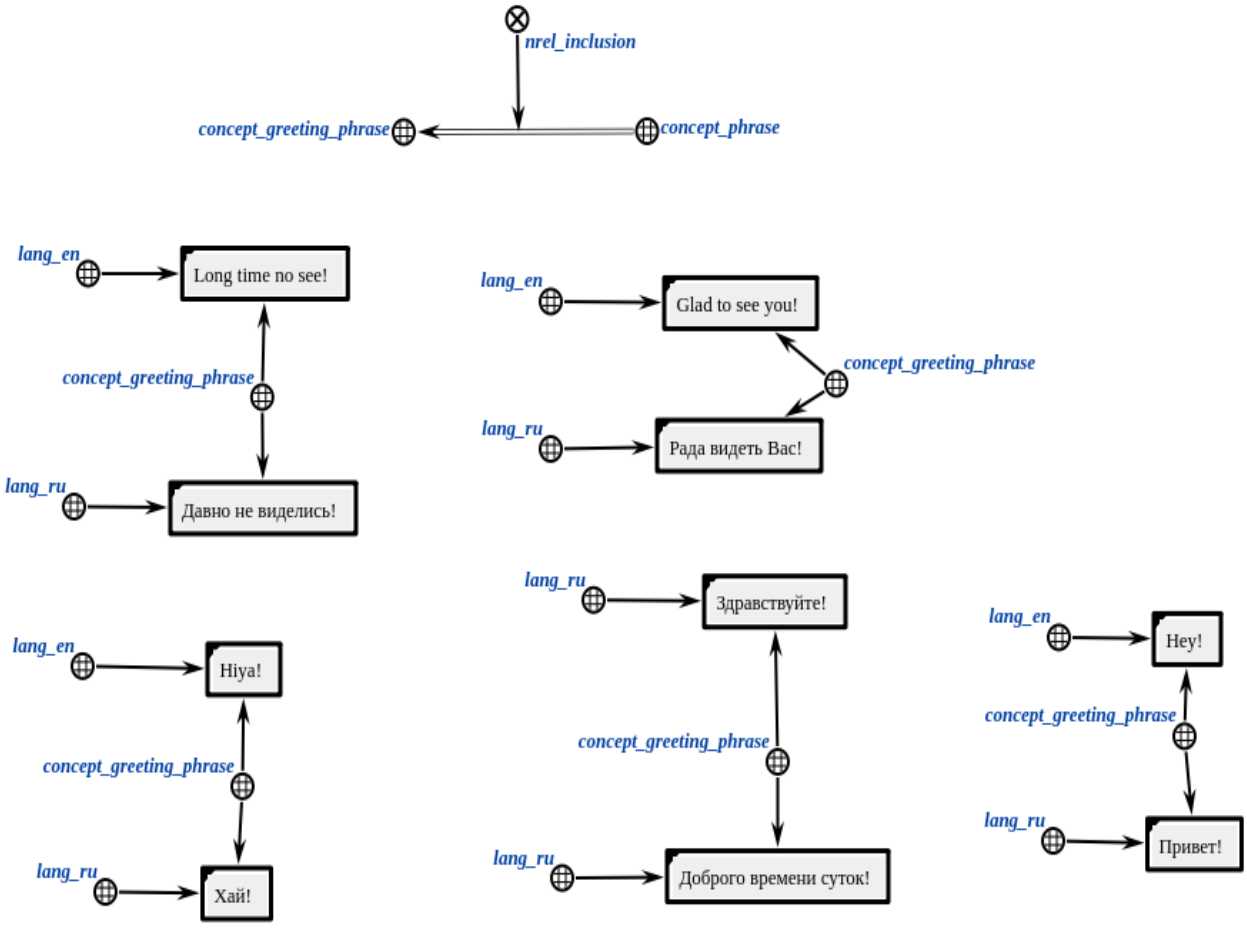

Рисунок 37 – Формализация фраз приветствия

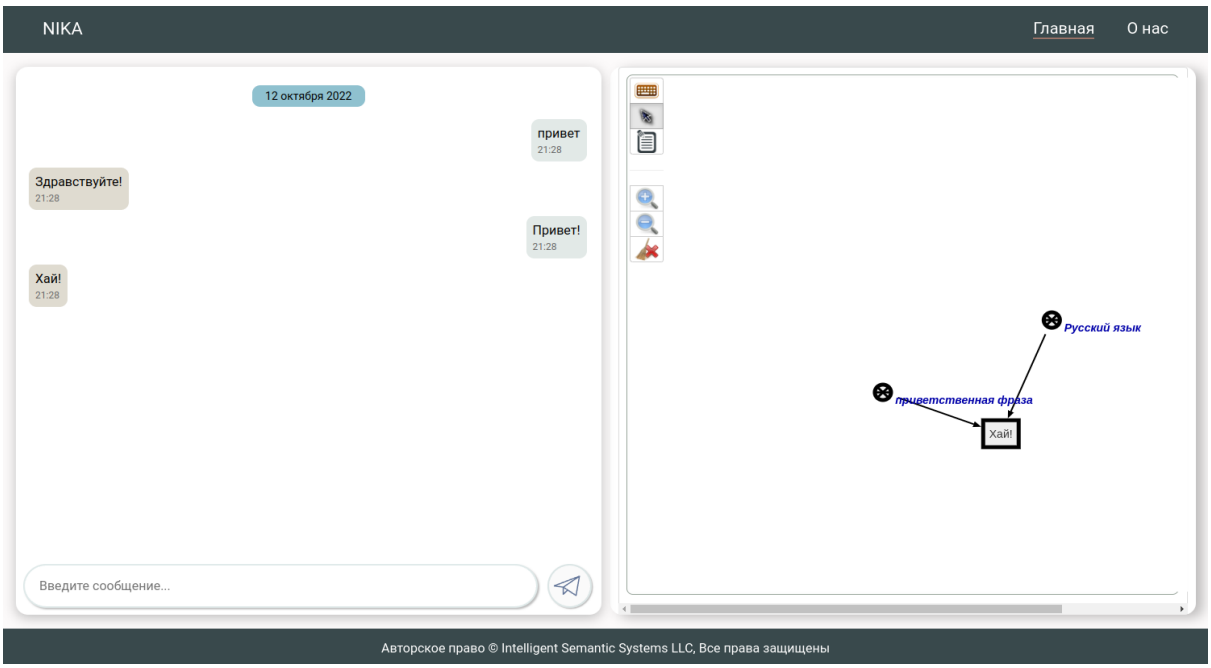

Рисунок 38 – Пример ответа на сообщения с помощью логических правил

Сообщение для ответа выбирается случайным образом из множества фраз того класса, который указан в шаблоне ответа логического правила.

Пример логической формулы с использованием сущности, выделяемой в сообщении, представлен на рисунке 39.

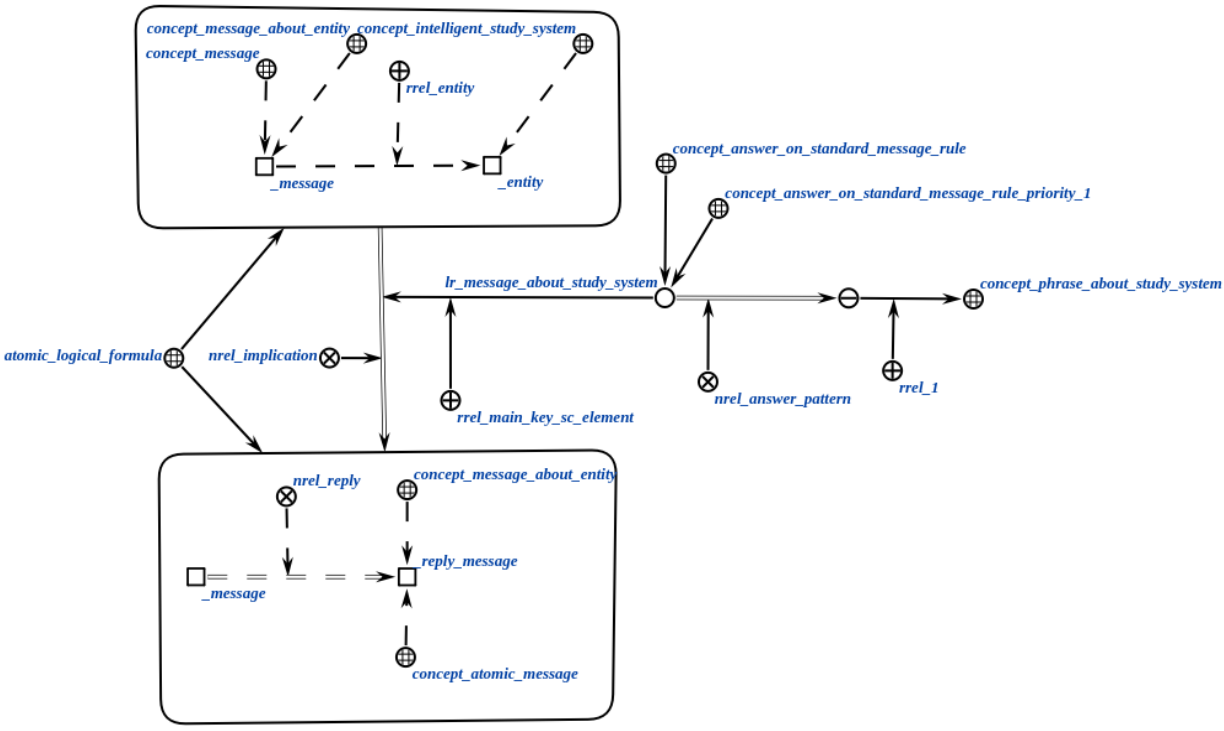

Рисунок 39 – Пример правила с использованием сущности, выделяемой в сообщении

В данном примере показано, что если появилось сообщение (*concept_message*), классифицированное как сообщение о сущности (*concept_message_about_entity*) с выделенной сущностью (*rrel_entity*) *_entity*, которая принадлежит классу *concept_intelligent_study_system*, то генерируется ответное сообщение (связка *nrel_reply* от *_message* к *_reply_message*), принадлежащее классам *concept_message_about_entity* и *concept_atomic_message*. Ответной фразой по такому правилу может быть фраза с использованием шаблона, где конкретная сущность используется в ответе.

Формализация фраз ответа. Формализация фраз для ответа на классифицируемое сообщение представлена на рисунке 40. При наличии нескольких фраз, принадлежащих одному классу, система для ответа выбирает любую фразу случайным образом.

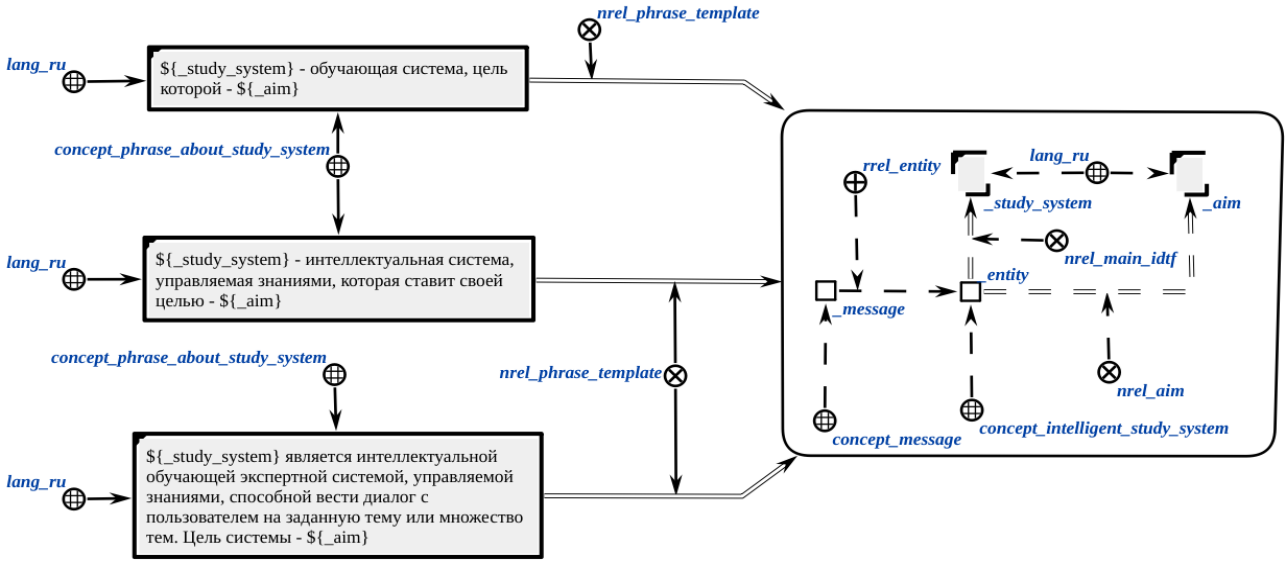

Рисунок 40 – Формализация фразы про обучающую систему с использованием шаблона

В данном примере показана формализация фразы ответа с использованием переменных.

Подготовка к защите

1 Протестировать диалог, используя спроектированные правила. Убедиться, что каждое правило может быть применено.

2 Протестировать правила и фразы в базе знаний системы (навигация по sc-web).

3 Сделать отчет (электронная версия) с иллюстрацией тестирования.

4 Загрузить изменения в свой форк на GitHub со всеми выполненными заданиями.

5 Подготовиться к ответу на вопросы и решению задач. Повторить теоретический материал.

Ход защиты

1 Проверка отчета.

2 Иллюстрация и тестирование правил путем получения ответа на сообщение.

3 Ответы на вопросы преподавателя и решение задач.

4 Выставление оценки.

Вопросы для подготовки к защите

1 Что такое формальная теория? Приведите пример.

2 Что такое высказывание? Какими они бывают?

3 В чем отличие фактографического высказывания от логической формулы?

4 Какие бывают логические связки?

5 В каком случае высказывание является бессмысленным?

6 Чем можно заменить конъюнкцию двух атомарных формул?

7 Что такое шаблон, для чего он нужен? Какие операции возможны с шаблонами?

8 Какие кванторы накладываются по умолчанию на атомарные логические формулы и на импликацию?

Задачи

1 Формализовать логические правила:

(1) Если текущее время на телефоне больше 23 часов или меньше 8 часов, то телефон в беззвучном режиме.

(2) Если температура воды не меньше 100 градусов по Цельсию, вода закипает.

(3) Если сообщение пришло после того, как рабочий день закончился, ответа на сообщение не будет.

(4) Если человек _x является предком человека _y и человек _y является предком человека _z, то человек _x является предком человека _z.

(5) Если некоторое яблоко или красное, или невкусное и это яблоко или зеленое, или вкусное, то оно красное или зеленое.

(6) Если студент ИИ понял главную проблему современного искусственного интеллекта, то он проучился не зря.

(7) Нет дыма без огня.

(8) Без труда не выловишь и рыбку из пруда.

(9) Дважды в год лето не бывает.

(10) Если гардероб работает и у студента идет занятие, куртка студента должна быть сдана в гардероб.

Пример решения задачи (1) представлен на рисунке 41.

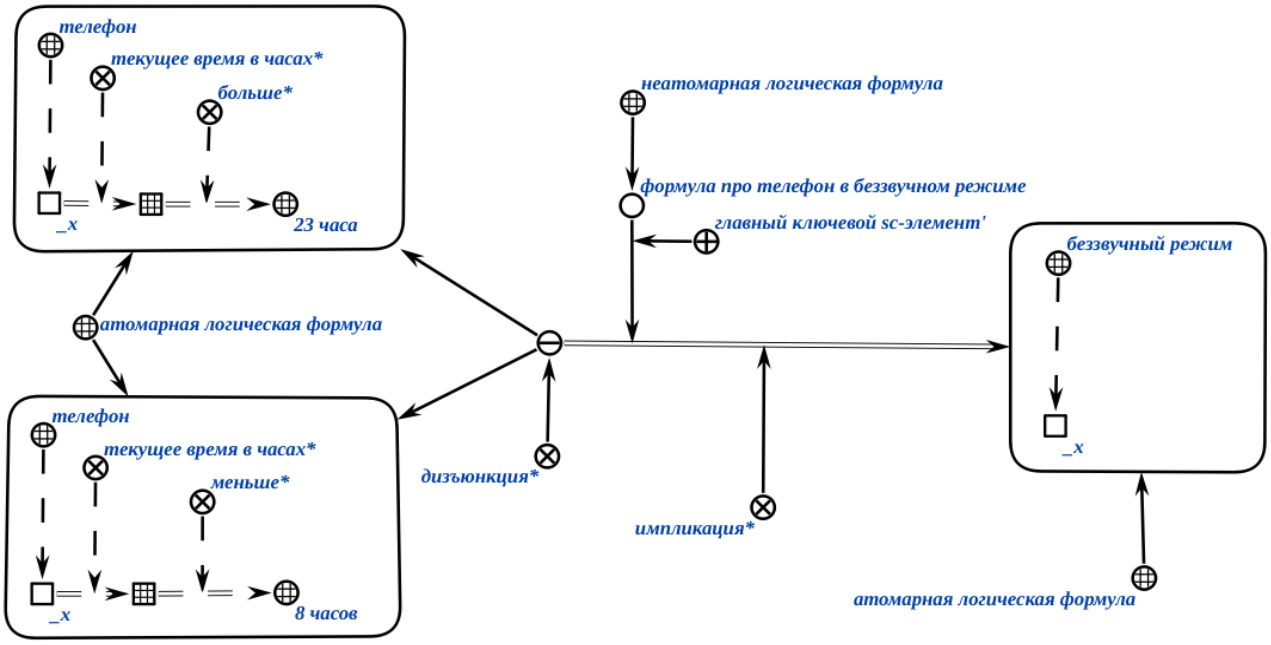

Рисунок 41 – Пример решения задачи «Если текущее время на телефоне больше 23 часов или меньше 8 часов, то телефон в беззвучном режиме»

2 Формализовать фразу с использованием шаблона:

(1) В 9 «В» классе учится 25 учеников, из которых 14 отличников, 7 хорошистов и 4 двоечника. Класс 9 «В» занял второе место в рейтинге классов школы. Класс 9 «В» углубленно изучает математику.

(2) Студент Иванов учится на 4-м курсе, имеет средний балл 8,1, живет в съемной квартире, работает в компании Google.

(3) В аудитории 612 находится 22 стола, 15 компьютеров. Ответственный за пожарную безопасность Петров.

(4) БГУИР основан в 1964 году, корпус № 1 находится по адресу ул. П. Бровки, 6. Количество учащихся 17 000, специализация университета – информационные технологии, радиотехника, электроника и телекоммуникации.

(5) Треугольник ABC имеет периметр 12 см, площадь 6 см^2. Тип треугольника ABC – прямоугольный. Соотношение сторон 3 : 4 : 5.

Пример решения задачи (5) представлен на рисунке 42.

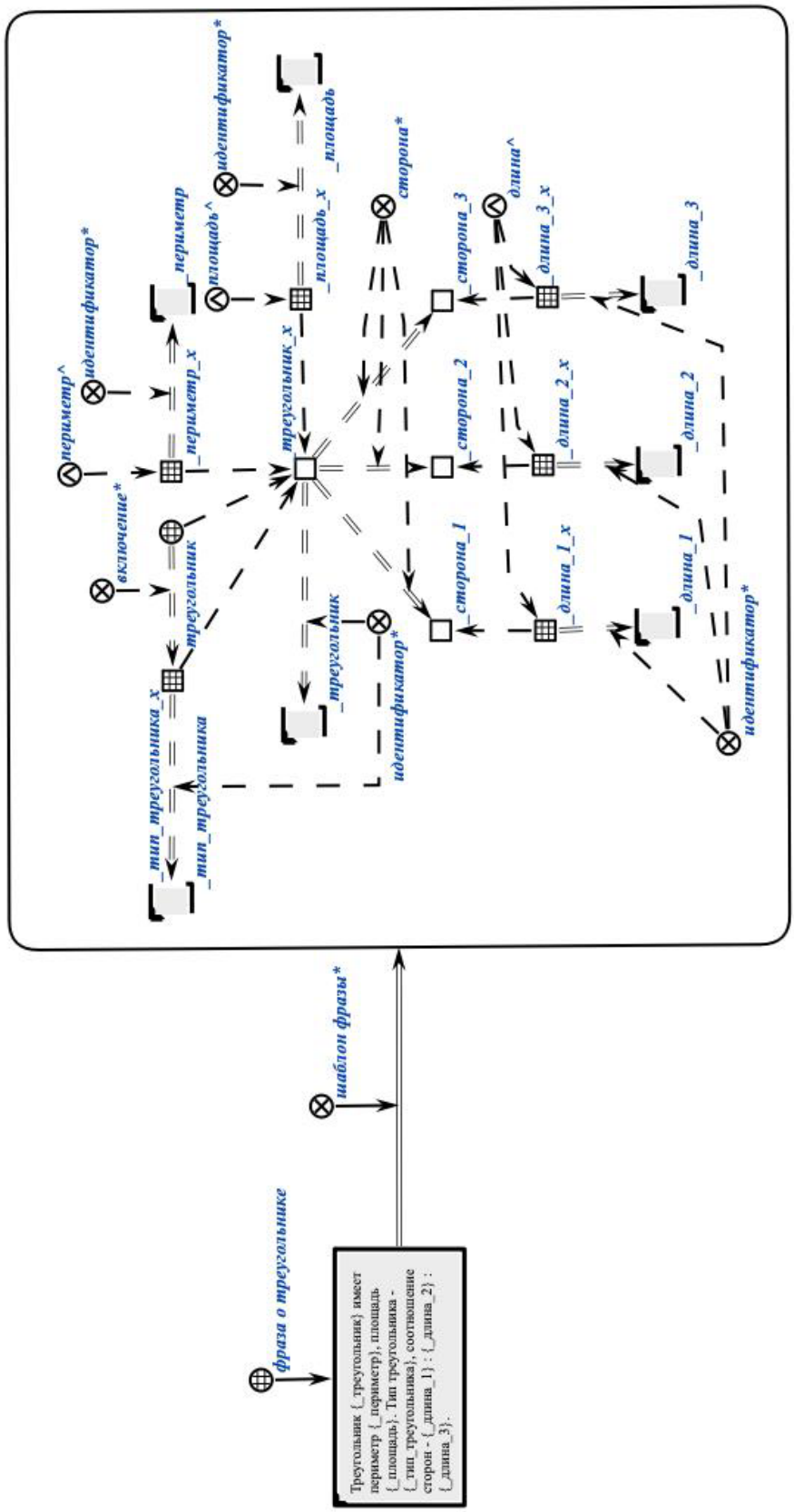

Рисунок 42 – Пример решения задачи «Треугольник ABC имеет периметр 12 см, площадь 6 см^2. Тип треугольника ABC – прямоугольный. Соотношение сторон 3 : 4 : 5»

Лабораторная работа № 4. Разработка модуля адаптации знаний для системы NIKA[1]

Цель: изучить принципы работы и проектирования решателей задач интеллектуальных систем, освоить навыки проектирования и реализации программ с помощью различных видов представления знаний и моделей решения задач, научиться анализировать и выводить знания.

Задание

1 Выбрать вариант (вариант должен быть уникален в рамках потока), также можно предложить свою идею реализации агента, которой нет в подготовленных вариантах (в этом случае необходимо согласовать свой вариант с преподавателем).

2 Спроектировать и реализовать агент согласно выбранному варианту. Агент должен уметь:

− получать знания из внешних или локальных источников и погружать в базу знаний;

− обрабатывать знания из локальных или внешних источников и выводить новые;

− выводить новые знания на основе знаний, погруженных в базу знаний.

3 Интегрировать агент в общую программу обработки системы.

4 Протестировать работу агента при помощи диалога с системой.

5 Задокументировать реализацию агента:

− специфицировать условие инициирования и результат работы агента;

− описать входы и выходы агента;

− описать функционал (процедуры) агента;

− составить схему алгоритма работы агента.

6 По результату работы составить отчет, в котором представить:

− цель выполненной работы;

− задание лабораторной работы;

− индивидуальный вариант;

− документацию из пункта 5;

− скриншоты функционала агента;

[1] Лабораторная работа № 4 не является обязательной для выполнения.

− иллюстрацию работы агента;

− выводы.

7 Уметь отвечать на вопросы и решать задачи, приведенные в конце лабораторной работы.

8 Загрузить изменения в свой форк на GitHub.

Варианты задания для выполнения лабораторной работы

Вариантом данной лабораторной работы должен являться вектор экземпляров выбранных категорий.

Класс 1. Локальность используемых источников:

1) локальные источники (база знаний системы NIKA, локальные структуры данных, файлы);

2) внешние сервисы, доступные по ссылке в сети Интернет;

3) веб-страницы (скраперы).

Класс 2. Структура внешней информации:

1) структурированные:

– excel;

– sql;

– graph-ql;

– rdf/owl;

– erp;

2) полуструктурированные:

– xml;

– json;

– html;

– no-sql;

– csv;

3) неструктурированные:

– txt;

– docs;

– pdf;

– image;

– audio.

Класс 3. Модель обработки знаний:

1) процедурные (алгоритмические) модели;

2) нейросетевые модели;

3) логические модели;

4) генетические алгоритмы;

5) интеграция нескольких моделей.

Класс 4. Язык программирования:

1) C++;

2) Python;

3) TypeScript.

Кроме того, можно выбрать вариант реализации агента из следующего списка:

1 Агент получения начальной формы для сущностей в классификации.

2 Агент получения конкретных знаний из диалога (узнать имя и использовать его в ходе диалога), т. е. агент должен уметь запоминать имя пользователя из диалога, использовать его во фразах.

3 Агент инициирования диалога.

4 Агент синтеза и распознавания речи.

5 Агент обработки чисел в Wit Response.

6 Агент обучения Wit.ai.

7 Агент с возможностью изменения языка ведения диалога, поддержкой нескольких языков одновременно.

8 Агент, который учитывает знания из предыдущих сообщений.

Теоретические сведения

В основе **решателя задач** каждой ostis-системы лежит многоагентная система, агенты которой взаимодействуют между собой *только* через общую для них sc-память посредством спецификации в этой памяти выполняемых ими действий в sc-памяти. При этом пользователи ostis-системы также считаются агентами этой системы. Кроме того, sc-агенты делятся на внутренние, рецепторные и эффекторные. Взаимодействие между агентами через общую sc-память сводится к следующим видам действий:

– использованию общедоступной для соответствующей группы sc-агентов части хранимой базы знаний;

– формированию (генерации) новых фрагментов базы знаний и/или корректировке (редактированию) каких-либо фрагментов доступной части базы знаний;

– интеграции (погружению) новых и/или обновленных фрагментов в состав доступной части базы знаний.

Подчеркнем, что sc-агенты не «общаются» между собой напрямую путем отправки сообщений, как это делается в большинстве современных подходов к построению многоагентных систем. Кроме того, sc-агенты имеют доступ к общей для них базе знаний, за счет чего гарантируется семантическая совместимость (взаимопонимание) между агентами, включая и пользователей ostis-систем.

Пользователь ostis-системы не может сам непосредственно выполнить какое-либо действие в sc-памяти, но он может средствами пользовательского интерфейса инициировать построение (генерацию, формирование в sc-памяти) sc-текста, являющегося спецификацией действия в sc-памяти, выполняемого либо одним атомарным sc-агентом за один акт, либо одним атомарным sc-агентом за несколько актов, либо коллективом sc-агентов (неатомарным sc-агентом). В спецификации каждого такого действия в sc-памяти, инициированного пользователем, пользователь указывается как заказчик этого действия. Таким образом, пользователь ostis-системы дает поручения (задания, команды) sc-агентам этой системы на выполнение различных специфицируемых им действий в sc-памяти.

Каждый sc-агент, выполняя некоторое действие в sc-памяти, должен «помнить», что sc-память, над которой он работает, является ресурсом не только для него, но и для всех остальных sc-агентов, работающих над ней. Поэтому sc-агент должен соблюдать определенную этику поведения в коллективе таких sc-агентов, чтобы минимизировать помехи, создаваемые другим sc-агентам.

Деятельность каждого агента ostis-системы дискретна и представляет собой множество элементарных действий (актов). При этом при выполнении каждого акта агент может устанавливать блокировки нескольких типов на фрагменты базы знаний. Данные блокировки позволяют запретить другим агентам изменять указанный фрагмент базы знаний или вообще сделать его невидимым для других агентов. Блокировки устанавливаются самим агентом

при выполнении соответствующего акта и снимаются им же на последнем этапе выполнения этого акта или раньше, если это возможно.

Таким образом, **sc-модель машины обработки знаний** любой ostis-системы представляет собой неатомарный абстрактный sc-агент, являющийся результатом объединения всех абстрактных sc-агентов, входящих в состав этой ostis-системы, в один. Другими словами, под sc-моделью машины обработки знаний понимается коллектив всех sc-агентов, входящих в состав заданной ostis-системы.

Под **агентом** понимается некоторый субъект действия (в частности процедура), который обладает такими качествами, как:

– автономность (независимость) агентов, что позволяет локализовать изменения, вносимые в систему при ее эволюции, и снизить соответствующие трудозатраты;

– децентрализация обработки, т. е. отсутствие единого контролирующего центра, что также позволяет локализовать вносимые в систему изменения;

– возможность параллельной работы информационного процесса, соответствующего этому агенту, т. е. возможность распределенного решения задач.

В системах, ориентированных на агентную обработку решаемых задач, сами задачи формулируются в декларативном ключе.

Ход работы

Программная реализация платформы интерпретации sc-моделей написана на языках C или C++ [14]. На данный момент существует два способа подключения к ней.

1 На уровне платформы предоставляется интерфейс, с помощью которого можно реализовать модули и агентные программы на языке C++, зависимые от реализации платформы. Под модулем подразумевается некоторый коллектив агентов, тесно связанных между собой (стоит отметить, что деление может быть и другим).

2 К sc-памяти можно подключиться по сети с помощью клиентов, подключенных к так называемому sc-серверу, работающему по websocket-протоколу. Такой вариант подключения независим от реализации платформы, но зависим от данных, которые передаются по сети в *json*-формате.

Разработка на языке C++

В качестве примера рассмотрим реализацию агента нахождения сущностей, начинающихся на определенную букву, на C++. Для реализации данного агента нужно выполнить следующие шаги.

1 В директории */nika/problem-solver/cxx* необходимо создать дополнительную директорию для реализации модуля и агентов этого модуля. Например, можно создать папку *messageProcessingModule*.

2 После этого в созданной директории необходимо подключить зависимости, а также указать исходники для сборки. Для указания зависимостей необходимо создать конфигурационный файл *CMakeLists.txt* и поместить в эту же директорию. Содержание может быть следующим:

```
# добавление всех исходных файлов C++ в списки для цели сборки
set(SOURCES
  "MessageProcessingModule.cpp"
  "keynodes/MessageProcessingKeynodes.cpp"
  "agent/FindWordInSetByFirstLetterAgent.cpp"
)

set(HEADERS
  "MessageProcessingModule.hpp"
  "keynodes/MessageProcessingKeynodes.hpp"
  "agent/FindWordInSetByFirstLetterAgent.hpp"
)

# включение исходных файлов sc-памяти
include_directories(
  ${CMAKE_CURRENT_LIST_DIR}
  ${SC_MEMORY_SRC}
  ${SC_KPM_SRC}
  ${DIALOG_CONTROL_MODULE_SRC}
  ${SC_COMMON_MODULE_SRC}
)

# создание библиотеки messageProcessingModule на базе собранных
исходных файлов
add_library(messageProcessingModule SHARED ${SOURCES} ${HEADERS})
# линковка зависимости messageProcessingModule библиотек от
```

```
# sc-memory и sc-agents-common
target_link_libraries(messageProcessingModule sc-memory sc-agents-
common common dialogControlModule)

# запуск генератора кода (необходим для создания тел
# sc-агентов и ключевых элементов)
sc_codegen_ex(messageProcessingModule    ${CMAKE_CURRENT_LIST_DIR}
${CMAKE_CURRENT_LIST_DIR}/generated)
```

3 После подключения библиотек *sc-machine* [14] необходимо прописать логику регистрации модуля агентов. Для этого необходимо создать класс модуля *MessageProcessingModule* со следующим содержанием:

– файл *MessageProcessingModule.hpp:*

```
#pragma once

#include "sc-memory/sc_memory.hpp"
#include "sc-memory/sc_module.hpp"

#include "keynodes/MessageProcessingKeynodes.hpp"
#include "agent/FindWordInSetByFirstLetterAgent.hpp"
#include "utils/ActionUtils.hpp"

#include "MessageProcessingModule.generated.hpp"

namespace messageProcessingModule
{
 class MessageProcessingModule : public ScModule
 {
  SC_CLASS(LoadOrder(100))
  SC_GENERATED_BODY()

  // Метод инициализации модуля и его агентов
  virtual sc_result InitializeImpl() override;

  // Метод деинициализации модуля и его агентов
  virtual sc_result ShutdownImpl() override;
 };

} // namespace messageProcessingModule
```

– файл *MessageProcessingModule.cpp:*

```
#include "MessageProcessingModule.hpp"

namespace messageProcessingModule
{
 // Регистрация модуля
 SC_IMPLEMENT_MODULE(MessageProcessingModule)

 sc_result MessageProcessingModule::InitializeImpl()
 {
  if (!MessageProcessingKeynodes::InitGlobal())
  return SC_RESULT_ERROR;

  // Регистрация агентов модуля
  ScMemoryContext                        ctx(sc_access_lvl_make_min,
  "MessageProcessingModule");
  if                        (ActionUtils::isActionDeactivated(&ctx,
  MessageProcessingKeynodes::action_find_word_in_set_by_first_lett
  er))
  {
   SC_LOG_ERROR("action_letter_search is deactivated");
  }
  else
  {
   SC_AGENT_REGISTER(FindWordInSetByFirstLetterAgent);
  }

  return SC_RESULT_OK;
 }

 sc_result MessageProcessingModule::ShutdownImpl()
 {
  // Дерегистрация агентов модуля
  SC_AGENT_UNREGISTER(FindWordInSetByFirstLetterAgent);
  return SC_RESULT_OK;
 }

} // namespace messageProcessingModule
```

4 Реализация необходимого функционала производится на уровне агента. Агент представляет собой асинхронно-вызываемую программу, решающую определенный класс задач и имеющую описанную в формальном виде спецификацию. В спецификации агента указываются: класс задач, решаемых агентом, первичное условие инициирования, условие инициирования этого агента и его целевой результат, ключевые элементы (отношения и классы), используемые агентом, и другое.

Перед началом написания реализации агента необходимо сформулировать решаемый им класс задач. Агент *FindWordInSetByFirstLetterAgent* умеет находить в множестве сущности, начинающиеся на определенную букву. Таким образом, действие агента может быть задано как *action_find_word_in_set_by_first_letter*, что означает класс таких действий, которые могут быть выполнены некоторым агентом поиска сущностей на определенную букву при наличии соответствующего условия в памяти системы. При этом сам экземпляр действия агента должен принадлежать классу всех возможных действий *question*.

Поскольку агентная программа является асинхронной, то важно задать условие инициирования и результат агента. Агент считается инициированным тогда и только тогда, когда была создана, например, выходящая константная позитивная постоянная дуга принадлежности от класса *question_initiated* к sc-узлу действия этого агента.

5 Логика агента может описываться в любом стиле программирования и представляет собой класс, описываемый в файлах формата *.hpp* и *.cpp*:

– *FindWordInSetByFirstLetterAgent.hpp:*

```
#pragma once

#include "sc-memory/kpm/sc_agent.hpp"
#include "sc-agents-common/keynodes/coreKeynodes.hpp"

#include "FindWordInSetByFirstLetterAgent.generated.hpp"
#include "searcher/MessageSearcher.hpp"

namespace messageProcessingModule
{
 class FindWordInSetByFirstLetterAgent : public ScAgent
```

```
  {
   // Тип события условия инициирования sc-агента - появление
   в sc-памяти
   // выходящей дуги из question_initiated к действию
   SC_CLASS(Agent,
   Event(scAgentsCommon::CoreKeynodes::question_initiated,
   ScEvent::Type::AddOutputEdge))

   SC_GENERATED_BODY();

   private:
   bool checkActionClass(ScAddr const & actionAddr);

   ScAddrVector createAnswer(std::string const & linkContent) const;

   std::string getMessageText(ScAddr const & messageAddr) const;

   std::unique_ptr<class      dialogControlModule::MessageSearcher>
   messageSearcher;
  };
}  // namespace messageProcessingModule
```

– *FindWordInSetByFirstLetterAgent.cpp:*

```
// Подключение полезных утилит из sc-machine
#include "sc-agents-common/utils/AgentUtils.hpp"
#include "sc-agents-common/utils/CommonUtils.hpp"
#include "sc-agents-common/utils/IteratorUtils.hpp"
#include "sc-agents-common/keynodes/coreKeynodes.hpp"
#include "MessageKeynodes.generated.hpp"

#include "keynodes/MessageProcessingKeynodes.hpp"
#include "keynodes/MessageKeynodes.hpp"

#include "FindWordInSetByFirstLetterAgent.hpp"

using namespace scAgentsCommon;
using namespace utils;

namespace messageProcessingModule
```

```
{
SC_AGENT_IMPLEMENTATION(FindWordInSetByFirstLetterAgent)
{
 // Проверка условия инициирования sc-агента
 ScAddr const & actionAddr = otherAddr;
 if (!checkActionClass(actionAddr))
 {
  return SC_RESULT_OK;
 }

 // Вывод сообщения о начале работы агента
 SC_LOG_DEBUG("FindWordInSetByFirstLetterAgent started");

 ScAddr const & messageAddr =
 utils::IteratorUtils::getAnyByOutRelation(&m_memoryCtx,
 actionAddr, scAgentsCommon::CoreKeynodes::rrel_1);
 if (!messageAddr.IsValid())
 {
  SC_LOG_ERROR("FindWordInSetByFirstLetterAgent: the message isn't
  valid");
  utils::AgentUtils::finishAgentWork(&m_memoryCtx, actionAddr,
  false);
  return SC_RESULT_ERROR;
 }

 if (!m_memoryCtx.HelperCheckEdge(
     MessageProcessingKeynodes::concept_message_about_find_word_by
     _first_letter,
     messageAddr,
     ScType::EdgeAccessConstPosPerm))
 {
  SC_LOG_DEBUG("FindWordInSetByFirstLetterAgent: the message isn't
  about letter search");
  utils::AgentUtils::finishAgentWork(&m_memoryCtx, actionAddr,
  false);
  return SC_RESULT_ERROR;
 }

 ScAddrVector answerElements;
 try
```

```
{
 ScIterator3Ptr const & agentAnswerLinkIterator =
 m_memoryCtx.Iterator3(
    MessageProcessingKeynodes::word_starts_with_required_letter_a
    nswer_phrase,
    ScType::EdgeAccessConstPosPerm,
    ScType::LinkConst);
 if (agentAnswerLinkIterator->Next())
 {
  m_memoryCtx.EraseElement(agentAnswerLinkIterator->Get(2));
 }

 messageSearcher =
 std::make_unique<dialogControlModule::MessageSearcher>(&m_memory
 Ctx);
 std::string messageText = getMessageText(messageAddr);
 ScAddr const & entityAddr =
 utils::IteratorUtils::getAnyByOutRelation(
  &m_memoryCtx, messageAddr,
  dialogControlModule::MessageKeynodes::rrel_entity);
 // Создание итератора для поиска элементов, у которых есть связь
 с entity
 ScIterator3Ptr const & entityNodesIterator =
  m_memoryCtx.Iterator3(entityAddr,
  ScType::EdgeAccessConstPosPerm, ScType::NodeConst);

 std::string firstLetter;
 std::string firstWordLetter;

 firstLetter = messageText.substr(messageText.size() - 3, 2);
 std::stringstream resultStream;
 std::string result;
 std::string word;

 if (entityNodesIterator->Next())
 {
  word = utils::CommonUtils::getMainIdtf(
          &m_memoryCtx, entityNodesIterator->Get(2),
          {scAgentsCommon::CoreKeynodes::lang_ru});
```

```
    firstWordLetter = word.substr(0, 2);
    if (firstLetter == firstWordLetter)
    {
     resultStream << word;
     SC_LOG_DEBUG("FindWordInSetByFirstLetterAgent: found word "
     << word);
    }
   }
   while (entityNodesIterator->Next())
   {
    word = utils::CommonUtils::getMainIdtf(
           &m_memoryCtx, entityNodesIterator->Get(2),
           {scAgentsCommon::CoreKeynodes::lang_ru});
    firstWordLetter = word.substr(0, 2);
    if (firstLetter == firstWordLetter)
    {
     if (resultStream.str().size() == 0)
      resultStream << word;
     else
      resultStream << ", " << word;
     SC_LOG_DEBUG("FindWordInSetByFirstLetterAgent: found word "
     << word);
    }
   }

   if (resultStream.str().empty())
   {
    result = "Таких слов нет";
   }
   else
   {
    result = resultStream.str();
   }
   answerElements = createAnswer(result);
   SC_LOG_DEBUG("FindWordInSetByFirstLetterAgent: reply message is
   generated");
  }
  catch (utils::ScException const & exception)
  {
```

```
    SC_LOG_ERROR(exception.Description());
    utils::AgentUtils::finishAgentWork(&m_memoryCtx, actionAddr,
    answerElements, false);
    SC_LOG_DEBUG("FindWordInSetByFirstLetterAgent: finished with an
    error");
    return SC_RESULT_ERROR;
   }

   utils::AgentUtils::finishAgentWork(&m_memoryCtx, actionAddr,
   answerElements, true);

   SC_LOG_DEBUG("FindWordInSetByFirstLetterAgent finished");
   return SC_RESULT_OK;
}

bool FindWordInSetByFirstLetterAgent::checkActionClass(ScAddr
const & actionAddr)
{
   return m_memoryCtx.HelperCheckEdge(
      MessageProcessingKeynodes::action_find_word_in_set_by_first_l
      etter, actionAddr, ScType::EdgeAccessConstPosPerm);
}

ScAddrVector
FindWordInSetByFirstLetterAgent::createAnswer(std::string const &
linkContent) const
{
   ScAddr const & answerLink =
   m_memoryCtx.CreateLink(ScType::LinkConst);
   m_memoryCtx.SetLinkContent(answerLink, linkContent);
   ScAddr const & edgeAccessConstPosPerm = m_memoryCtx.CreateEdge(
    ScType::EdgeAccessConstPosPerm,
    MessageProcessingKeynodes::word_starts_with_required_letter_ans
    wer_phrase,
    answerLink);
   return {
    answerLink, edgeAccessConstPosPerm,
    MessageProcessingKeynodes::word_starts_with_required_letter_ans
    wer_phrase};
    }
```

```
 std::string
 FindWordInSetByFirstLetterAgent::getMessageText(ScAddr const &
 messageAddr) const
 {
  ScAddr const & messageLink = messageSearcher-
  >getMessageLink(messageAddr);
  if (!messageLink.IsValid())
  {
   SC_THROW_EXCEPTION(utils::ExceptionItemNotFound,
   "FindWordInSetByFirstLetterAgent: message link is not found.");
  }
  return utils::CommonUtils::getLinkContent(&m_memoryCtx,
  messageLink);
 }

}  // namespace messageProcessingModule
```

6 Также следует создать класс, содержащий необходимые сущности для работы агента:

– *MessageProcessingKeynodes.hpp:*

```
#pragma once

#include "sc-memory/sc_addr.hpp"
#include "sc-memory/sc_object.hpp"
#include "MessageProcessingKeynodes.generated.hpp"

namespace messageProcessingModule
{
 class MessageProcessingKeynodes : public ScObject
 {
  SC_CLASS()
  SC_GENERATED_BODY()

  public:
         SC_PROPERTY(Keynode("action_find_word_in_set_by_first_let
         ter"), ForceCreate)
         static ScAddr action_find_word_in_set_by_first_letter;
```

```
        SC_PROPERTY(Keynode("concept_message_about_find_word_by_f
        irst_letter"), ForceCreate)
        static                                                ScAddr
        concept_message_about_find_word_by_first_letter;

        SC_PROPERTY(Keynode("word_starts_with_required_letter_ans
        wer_phrase"), ForceCreate)
        static                                                ScAddr
        word_starts_with_required_letter_answer_phrase;
 };

} // namespace messageProcessingModule
```

– *MessageProcessingKeynodes.cpp:*

```
#include "MessageProcessingKeynodes.hpp"

namespace messageProcessingModule
{
 ScAddr MessageProcessingKeynodes::action_find_word_in_set_by_firs
 t_letter;
 ScAddr
 MessageProcessingKeynodes::concept_message_about_find_word_by_fir
 st_letter;
 ScAddr
 MessageProcessingKeynodes::word_starts_with_required_letter_answe
 r_phrase;
} // namespace messageProcessingModule
```

Стоит отметить, что программный код приводится в качестве примера, в реалиях логика и спецификация агента могут быть иными.

Как видно из листинга кода, агент в случае ошибки возвращает код ошибки, а в случае успешного выполнения – код успеха и ответную структуру, например структуру, которой принадлежит ссылка, включающая в себя найденные слова.

После реализации программного кода агента нужно совершить пересборку решателя задач. Для этого следует выполнить в *nika/scripts* скрипт *./build_problem_solver.sh* (либо *docker compose build* при использовании Docker).

При выполненной данной работы требуется использовать не только стандартные библиотеки ostis-платформы, но и сторонние библиотеки (например, библиотеку для обращения к API некоторого облачного сервиса).

Для того чтобы подключить разработанный агент в общую последовательность плана выполнения агентов системы NIKA, нужно изменить программу обработки (рисунок 43), которая находится на GitHub [20].

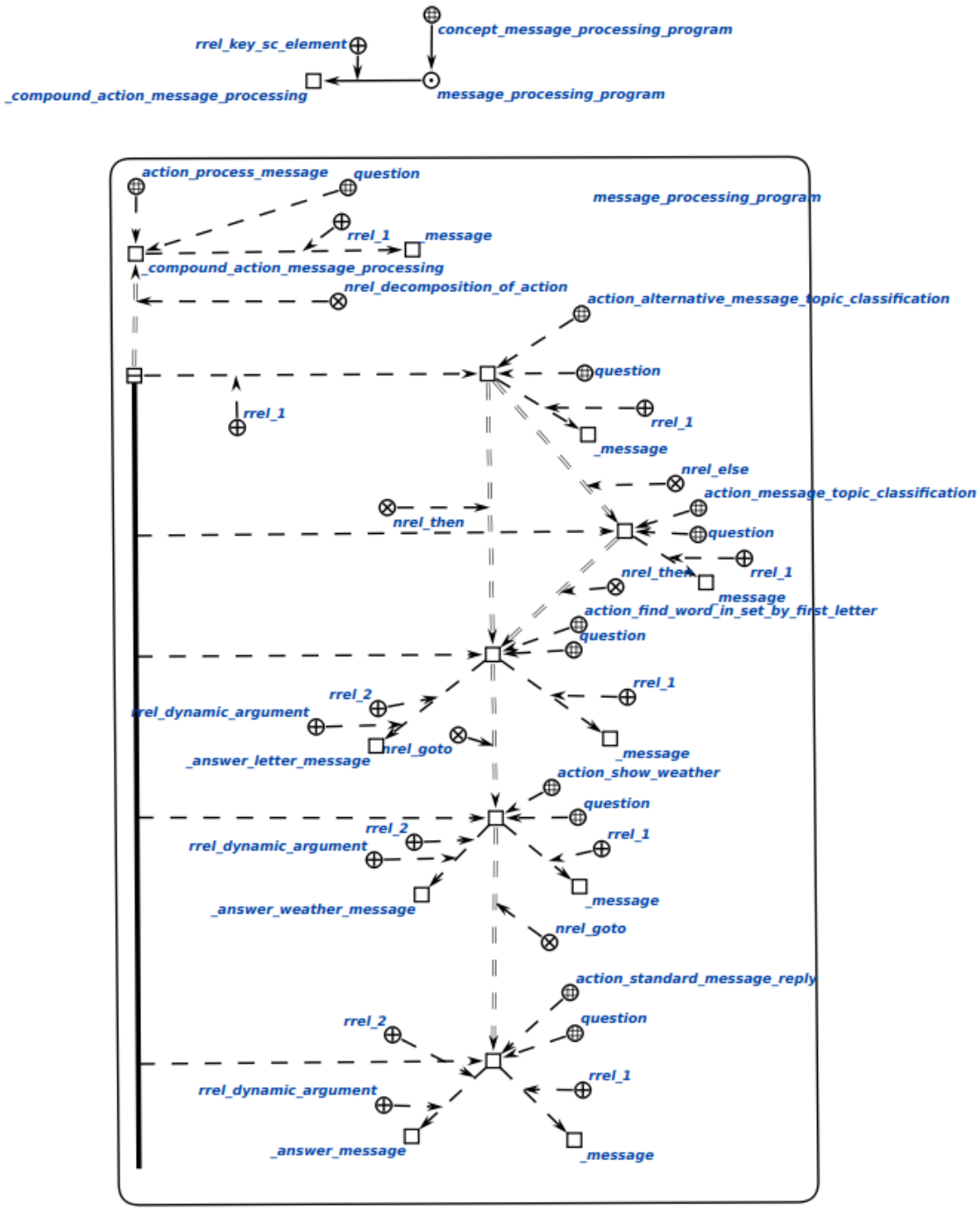

Рисунок 43 – Пример изменения программы обработки

Документацию агентов программы обработки системы NIKA можно найти также на GitHub [21].

Разработка на языках Python и TypeScript

Взаимодействовать с памятью ostis-системы можно не только при помощи программного API. Для решения более широкого круга задач была разработана подсистема взаимодействия с sc-памятью на основе WebSocket и JSON. Обработка запросов к памяти реализуется на стороне sc-сервера, являющегося частью sc-machine. Для обращения к sc-памяти через sc-сервер можно использовать клиенты, реализованные на разных языках. Больше информации о sc-сервере можно найти в Документации программного варианта реализации платформы интерпретации sc-моделей компьютерных систем [9].

Существует три открытых клиента, подключаемых к sc-серверу и удобных в использовании:

− клиент на Python – *py-sc-client* [22];

− машина обработки знаний на Python – *py-sc-kpm* [23];

− клиент на TypeScript – *ts-sc-client* [24].

Для разработки агента на языке Python в систему добавлен модуль и агент, который можно использовать в качестве примера [25].

Для разработки на TypeScript на данный момент рекомендуется самостоятельно реализовать подсистему с агентом, используя клиент для работы с памятью.

Подготовка к защите

1 Протестировать работоспособность агента через диалог с системой.

2 Сделать отчет (электронная версия) с иллюстрацией тестирования и документацией агента.

3 Загрузить изменения в форк на GitHub со всеми выполненными заданиями.

4 Подготовиться к ответу на вопросы. Повторить теоретический материал.

Ход защиты

1 Проверка отчета.

2 Иллюстрация и тестирование цепочки агентов.

3 Ответы на вопросы преподавателя.

4 Выставление оценки.

Вопросы для подготовки к защите

1 Что такое многоагентная система? Приведите пример.

2 Какие бывают многоагентные системы?

3 Зачем нужно создавать многоагентные системы? Возможно ли разрабатывать интеллектуальные системы, не являющиеся многоагентными?

4 Что такое интеллектуальная система?

5 Что такое агент? Какие бывают агенты?

6 Что входит в спецификацию любого агента?

7 Какие задачи позволяет решать текущая реализация платформы OSTIS?

ПРИЛОЖЕНИЕ А
Примеры взаимодействия с NIKA

Рассмотрим вопросы, на которые умеет отвечать NIKA, а также действия, которые необходимо выполнить для того, чтобы она смогла на них ответить.

Во-первых, чтобы узнать, на какие вопросы NIKA умеет отвечать, можно запустить проект. Вы увидите диалоговое окно. Введите пустое сообщение (нажмите Enter). В качестве сообщения-подсказки будет выбрано следующее: «Что Ника умеет?» (рисунок А.1). В ответе NIKA будут примеры вопросов, на которые она умеет отвечать.

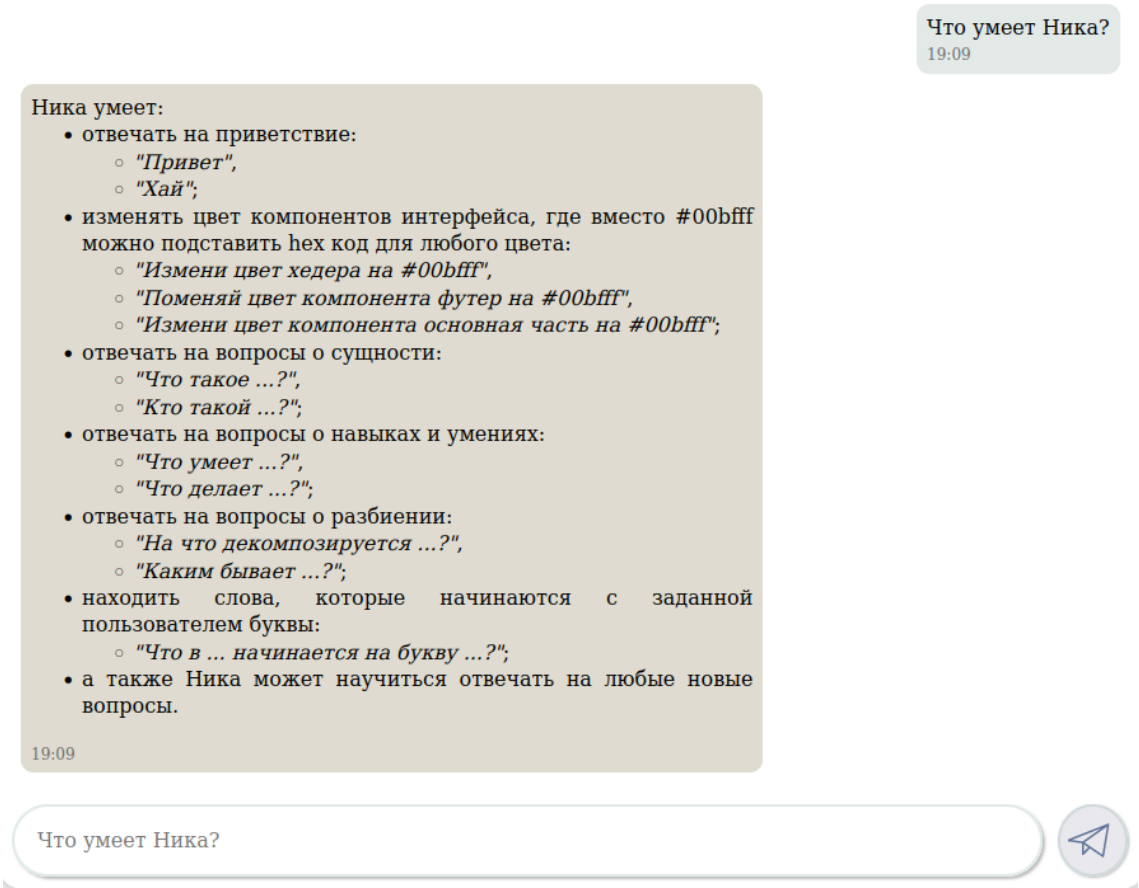

Рисунок А.1 – Ответ системы NIKA на вопрос «Что умеет NIKA?»

Для того чтобы NIKA смогла ответить на ваш вопрос, ей вначале нужно выделить *ключевые элементы сообщения* – сущности. Далее в своей базе знаний NIKA может найти семантическую окрестность данных сущностей и выделить *класс сообщения* (класс сообщения должен быть формализован в системе). Потом она по правилу *логического вывода* (Modus Ponens) попытается ответить на задаваемый

вопрос (фраза ответа на сообщение должна быть формализована в системе). В этом случае фраза может быть получена по атомарной логической формуле (шаблону), где для каждой sc-переменной будет подставлен в соответствие sc-элемент из семантической окрестности выделенной сущности. В результате данная фраза будет выведена в диалоговом окне как ответное сообщение.

Если NIKA не сможет выделить ключевой элемент сообщения (найти его семантическую окрестность), то в диалоговом окне появится ответная фраза о том, что NIKA не может ответить на вопрос (рисунок А.2).

Рисунок А.2 – Ответ системы NIKA в случае, если она не знает ответа на вопрос

При создании ответа на вопрос можно выделить следующие шаги:

1 Формализация необходимого класса сообщений.

2 Обучение Wit.ai отвечать на сообщения необходимого класса.

3 Формализация правила классификации сообщений необходимого класса.

4 Формализация правила ответа на сообщения необходимого класса.

5 Формализация фразы ответа на сообщения необходимого класса.

Под сообщением необходимого класса понимается вопрос (например, «Что NIKA умеет?»), использование слова «*класс*» указывает на то, что NIKA в итоге научится отвечать не только на конкретный вопрос («Что NIKA умеет?»), но и на схожие по формулировке вопросы («Что делает NIKA?», «Что NIKA может?», «Что может робот?» и т. д.).

Далее разберем формализацию следующих вопросов: «Что такое …?», «Что умеет …?», «На что декомпозируется …?» в соответствии с выделенными шагами.

«Что такое …?»

1 Данный класс сообщений лучше формализовать в отдельном файле с расширением *.scs*.

Название файла должно совпадать с названием класса сообщений, а сам файл должен быть расположен по следующему пути:

/nika/kb/section_subject_domain_of_messages/{папка с вашей ПрО}/concept_message_about_{название типа сообщения}.scs.

Ключевым элементом вопроса является какая-то сущность (не обязательно о NIKA), поэтому тип вопроса – о сущности *concept_message_about_entity.scs.*

Содержимое файла с формализацией класса можно увидеть в проекте:

```
concept_message_about_entity
<- sc_node_class;
=> nrel_main_idtf:
   [сообщение о сущности]
   (* <- lang_ru;; *);
   [message about entity]
   (* <- lang_en;; *);
=> nrel_wit_ai_idtf:
   [about_entity];
<- rrel_key_sc_element:
   ...
   (*
   => nrel_main_idtf:
      [Опр. (сообщение о сущности)]
      (* <- lang_ru;; *);
      [Def. (message about entity)]
      (* <- lang_en;; *);;
   <- definition;;
   <= nrel_sc_text_translation:
      ...
      (*
        -> rrel_example:
         [Сообщение о сущности - атомарное сообщение, которое
         содержит сущность, информацию о которой необходимо найти.]
         (* <- lang_ru;; *);;
    *);
    ...
    (*
    -> rrel_example:
       [Message about entity is an atomic message that contains
       entity information about which you need to find.]
       (* <- lang_en;; *);;
    *);;
    <= nrel_using_constants:
       {
        concept_atomic_message
       };;
*);;
```

Обучение на Wit.ai необходимо для классификации сообщений и выделения сущностей сообщения. Подробно данный этап рассмотрен в лабораторной работе № 2. Фразы для обучения Wit.ai могут быть следующими: «Что такое NIKA?», «Что NIKA такое?», «Кто такая NIKA?», «Что такое цветок?», «Кто такой космонавт?» и т. д.

2 Формулы (правила) классификации лучше всего формализовать на SCg-коде в файлах с расширением *.gwf* (рисунок А.3).

Название файла: *lr_message_about_{тип сообщения}.gwf*. Приставка «*lr*» означает «*logic rule*», т. е. «логическое правило» (логическая формула). Пример формализации можно увидеть по следующему пути: *nika/kb/section_subject_domain_of_dialogues/section_subject_domain_of_dialog_control/section_subject_domain_of_dialogue_control_using_template_responses/logic_rules/lr_alternative_classification/lr_message_about_entity.gwf*.

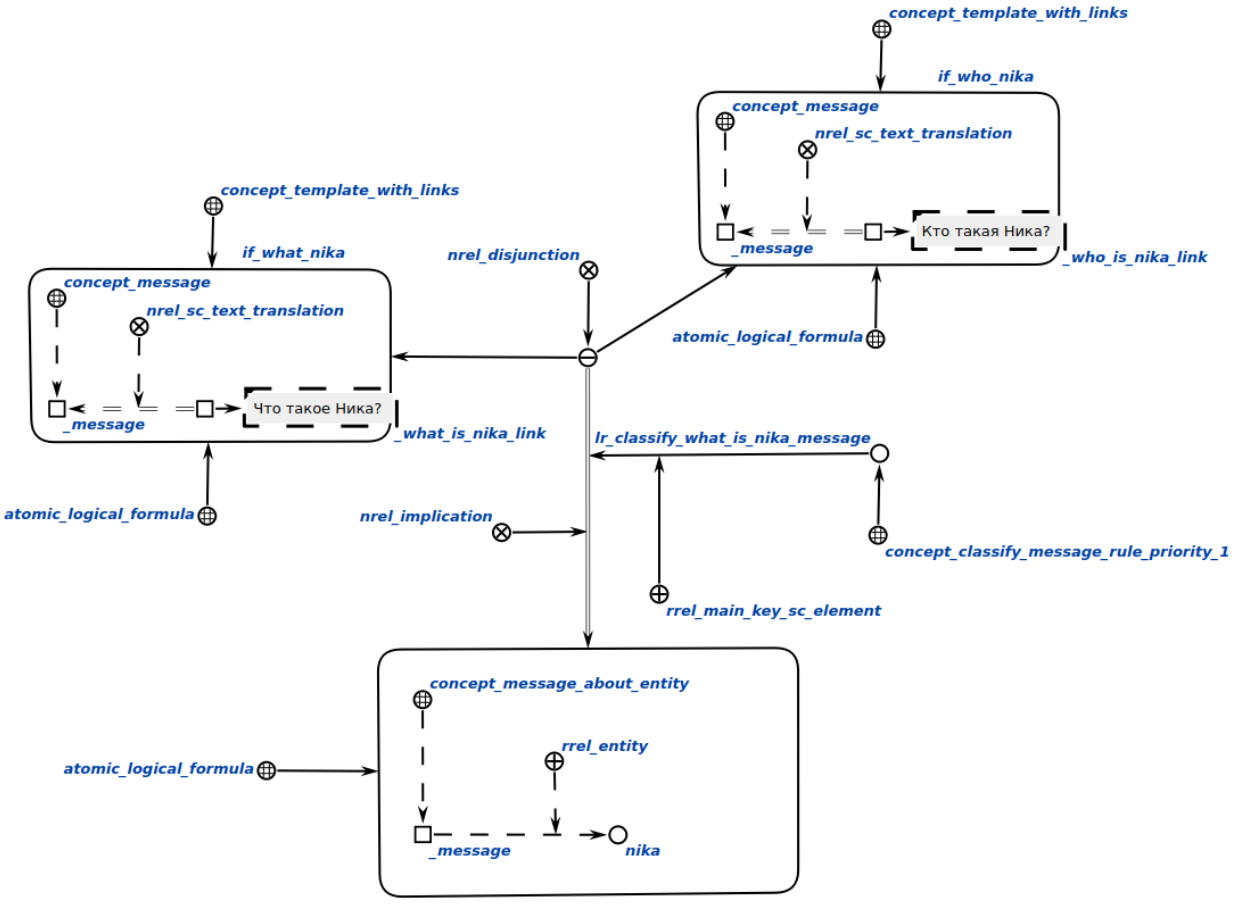

Рисунок А.3 – Пример формализации вопроса для системы NIKA

3 Формулы ответа на сообщения заданного класса, как и формулы классификации, лучше формализовывать на SCg-коде. Расположение примера:

nika/kb/section_subject_domain_of_dialogues/section_subject_domain_of_dialog_control/section_subject_domain_of_dialogue_control_using_template_responses/logic_rules/lr_message_about_entity/lr_message_about_entity.gwf.

Логическая формула представляет собой связку импликации между двумя логическими формулами, и данное правило необходимо для формирования ответа на сообщение, которое было успешно классифицировано (рисунок А.4).

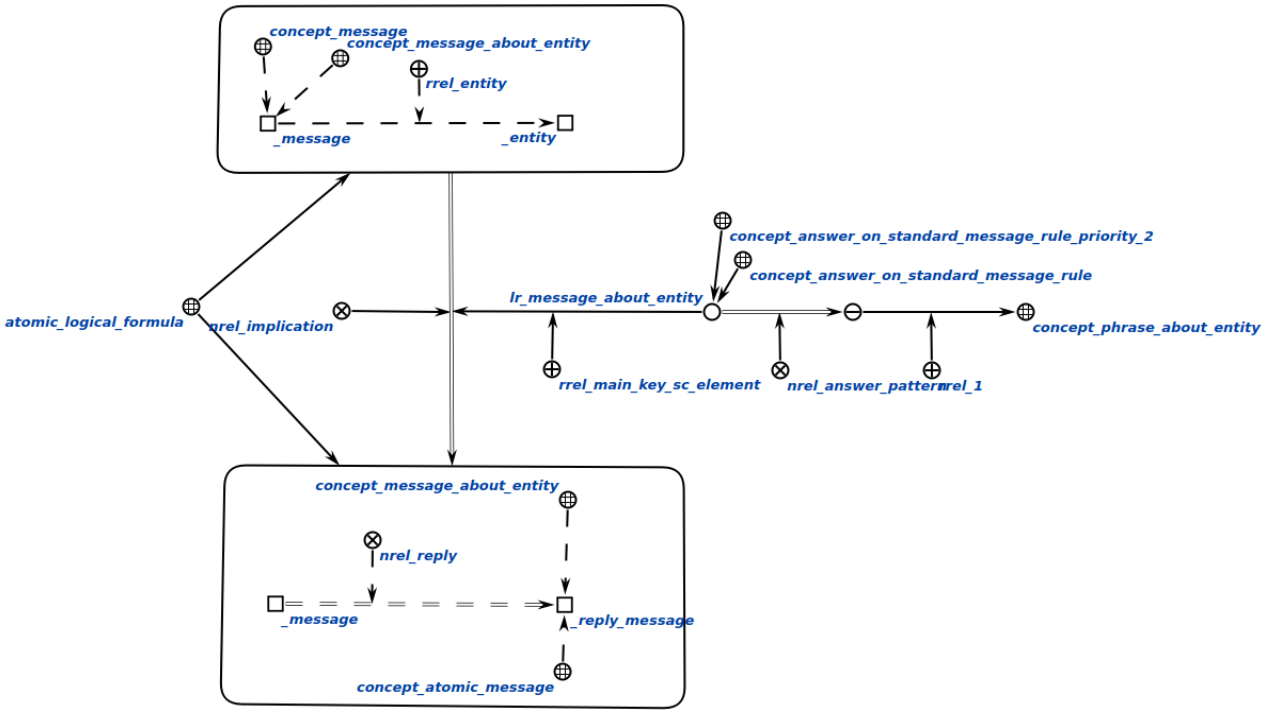

Рисунок А.4 – Пример логической формулы

4 Формулы генерации фразы ответа тоже лучше формализовать на SCg-коде (рисунок А.5). Расположение примера: *nika/kb/section_subject_domain_of_messages/subject_domain_of_phrases/examples/concept_phrase_about_entity/concept_phrase_about_entity.gwf.*

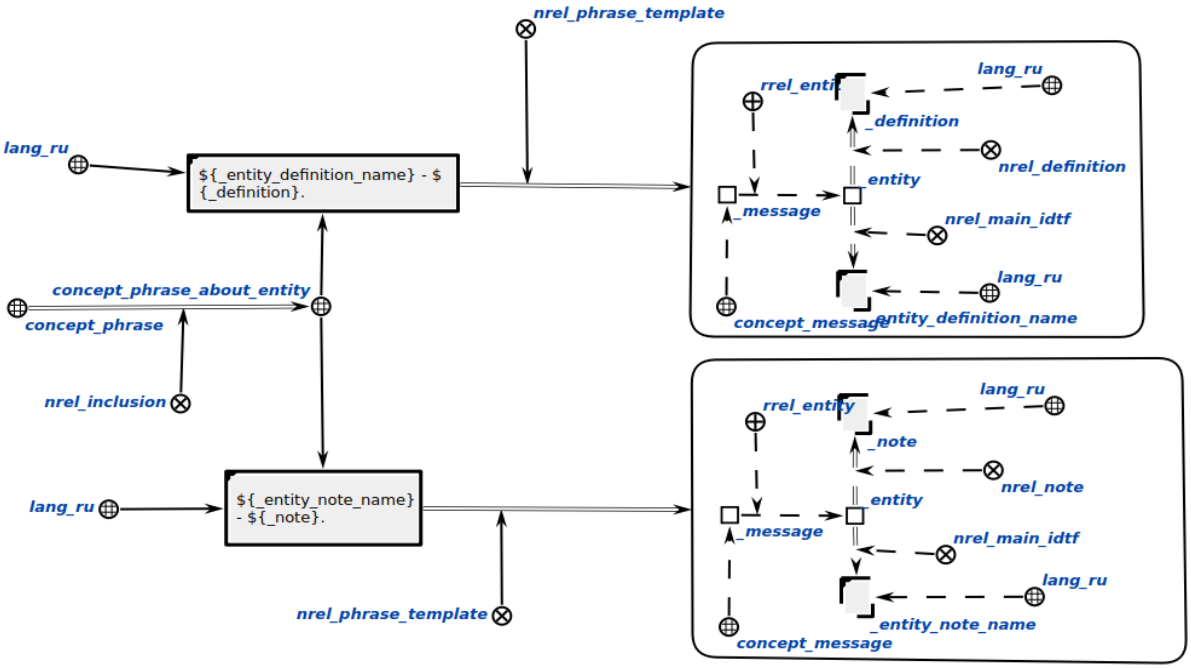

Рисунок А.5 – Пример генерации фразы ответа

«Что умеет …?»

1 Класс сообщений *concept_message_about_skill.scs*:

```
concept_message_about_skill
<- sc_node_class;
=> nrel_main_idtf:
   [сообщение о навыке]
    (* <- lang_ru;; *);
   [message about skill]
    (* <- lang_en;; *);
=> nrel_wit_ai_idtf:
   [about_skill];
<- rrel_key_sc_element:
   ...
   (*
   => nrel_main_idtf:
      [Опр. (сообщение о навыке)]
       (* <- lang_ru;; *);
      [Def. (message about skill)]
       (* <- lang_en;; *);;
   <- definition;;
   <= nrel_sc_text_translation:
       ...
```

```
        (*
        -> rrel_example:
           [Сообщение о навыке - атомарное сообщение, которое
           содержит навык, информацию о котором необходимо найти.]
            (* <- lang_ru;; *);;
        *);
        ...
        (*
        -> rrel_example:
           [Message about skill is an atomic message that contains
           skill information about which you need to find.]
            (* <- lang_en;; *);;
        *);;
    <= nrel_using_constants:
       {
        concept_atomic_message
       };;
    *);;
```

2 Фразы для обучения: «Что может NIKA?», «Что NIKA умеет?», «Что NIKA делает?», «Что может робот?» и т. д.

3 Формула классификации: *lr_message_about_skills.gwf* (рисунок А.6).

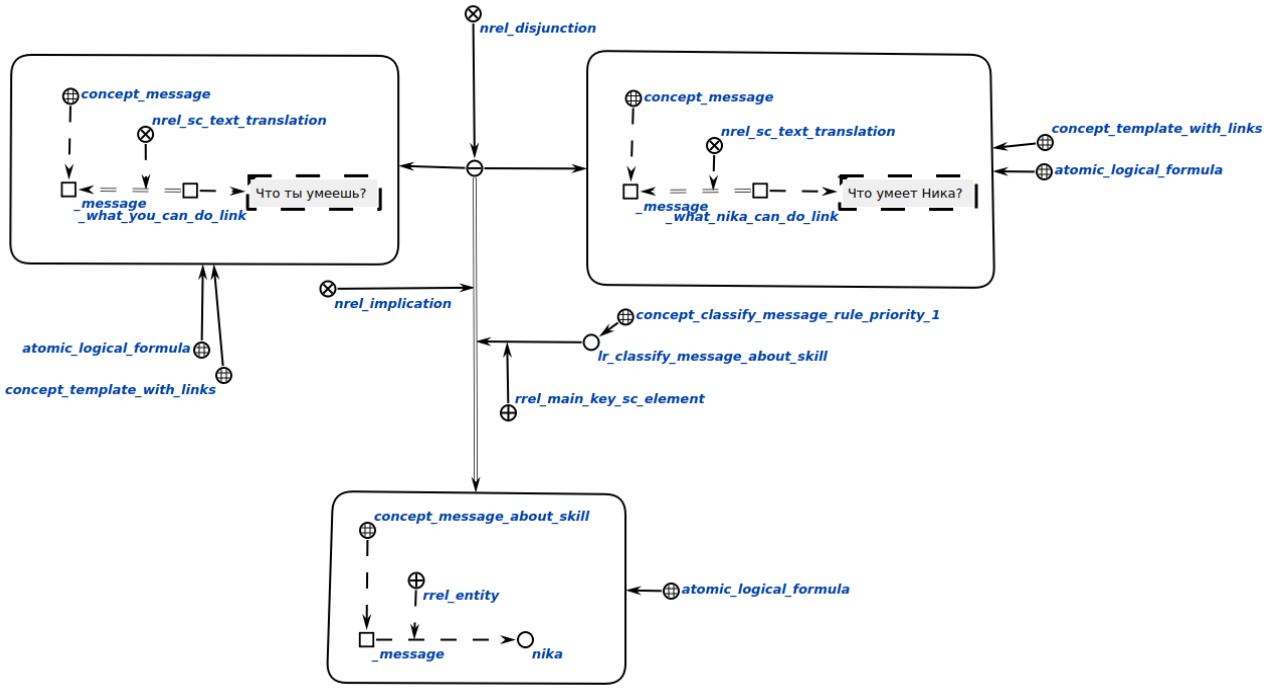

Рисунок А.6 – Пример формулы классификации

4 Формула ответа: *lr_message_about_skills.gwf* (рисунок А.7).

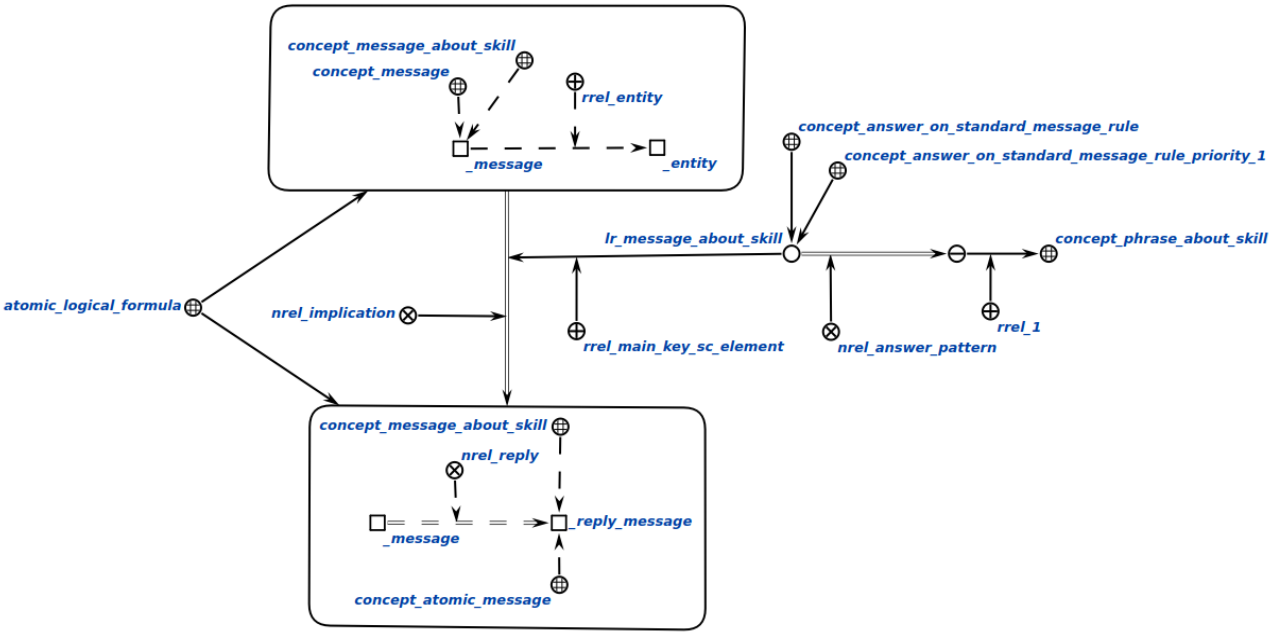

Рисунок А.7 – Пример формулы ответа на сообщение

5 Формулы генерации фразы ответа: *concept_phrase_about_skills.gwf* (рисунок А.8).

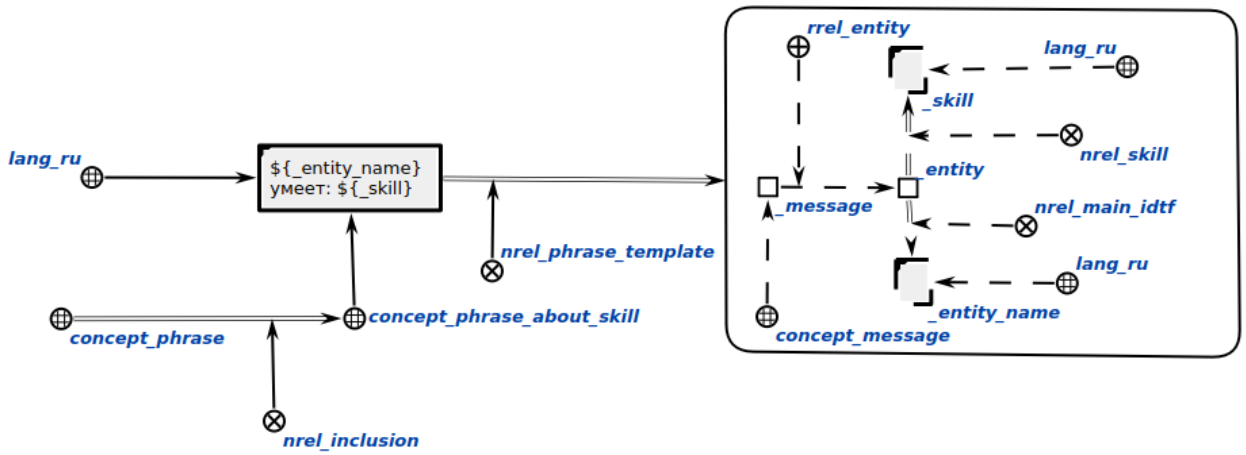

Рисунок А.8 – Пример формулы генерации фразы ответа

«На что декомпозируется …?»

1 Класс сообщений *concept_message_about_subdividing.scs*:

```
concept_message_about_subdividing
<- sc_node_class;
=> nrel_main_idtf:
   [сообщение о разбиении]
    (* <- lang_ru;; *);
```

```
    [message about subdividing]
     (* <- lang_en;; *);
 => nrel_wit_ai_idtf:
    [subdividing];
 <- rrel_key_sc_element:
    ...
    (*
      => nrel_main_idtf:
         [Опр. (сообщение о разбиении)]
          (* <- lang_ru;; *);
         [Def. (message about subdividing)]
          (* <- lang_en;; *);;
      <- definition;;
      <= nrel_sc_text_translation:
         ...
         (*
           [сообщение о разбиении - атомарное сообщение, в котором
           говорится о разбиении какой-то сущности.]
            (* <- lang_ru;; *);;
         *);
         ...

         (*
          -> rrel_example:
             [message about subdividing is an atomic message, that
             is about of an entity subdivision.]
              (* <- lang_en;; *);;
         *);;
         <= nrel_using_constants:
           {
            concept_atomic_message;
            nrel_subdividing
           };;
    *);;
```

2 Фразы для обучения: «Каким бывает сообщение?», «На что разбивается сообщение?», «Что может быть разбиением сообщения?», «Каким бывает действие?» и т. д.

3 Формула классификации: *lr_message_about_subdividing.gwf* (рисунок А.9).

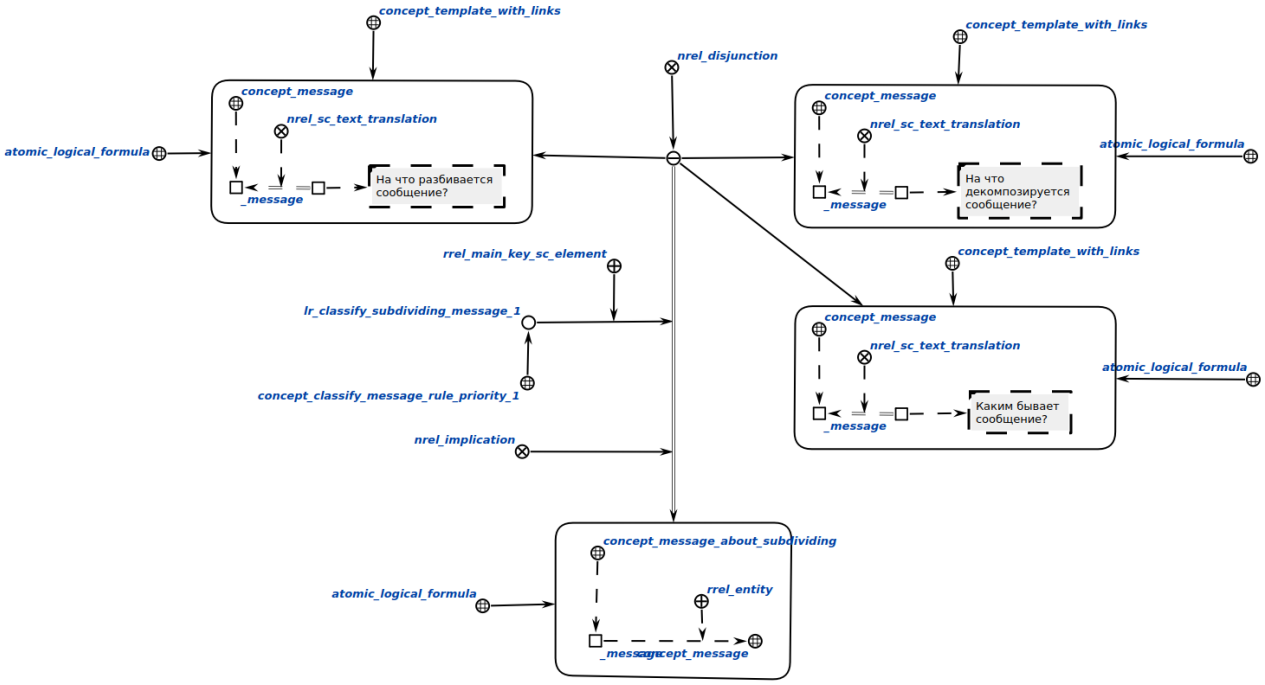

Рисунок А.9 – Пример формулы классификации

4 Формула ответа: *lr_message_about_subdividing.gwf* (рисунок А.10).

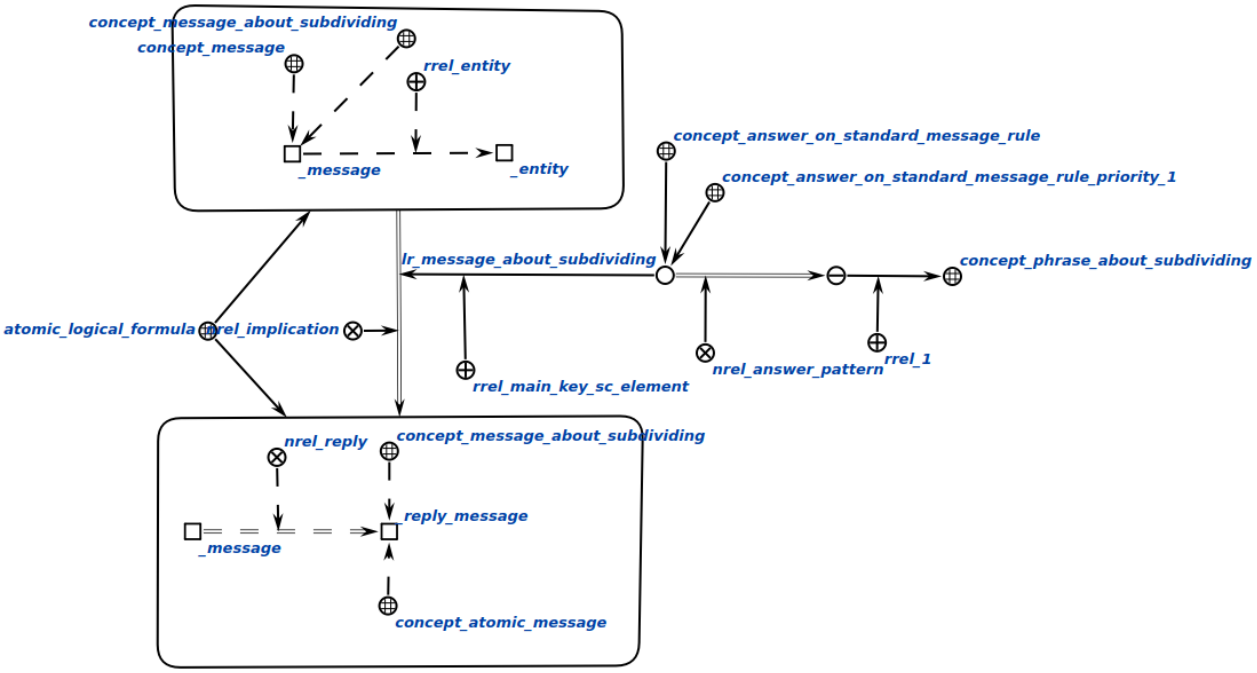

Рисунок А.10 – Пример формулы ответы для классификации

5 Формулы генерации фразы ответа: *concept_phrase_about_subdividing.gwf* (рисунок А.11).

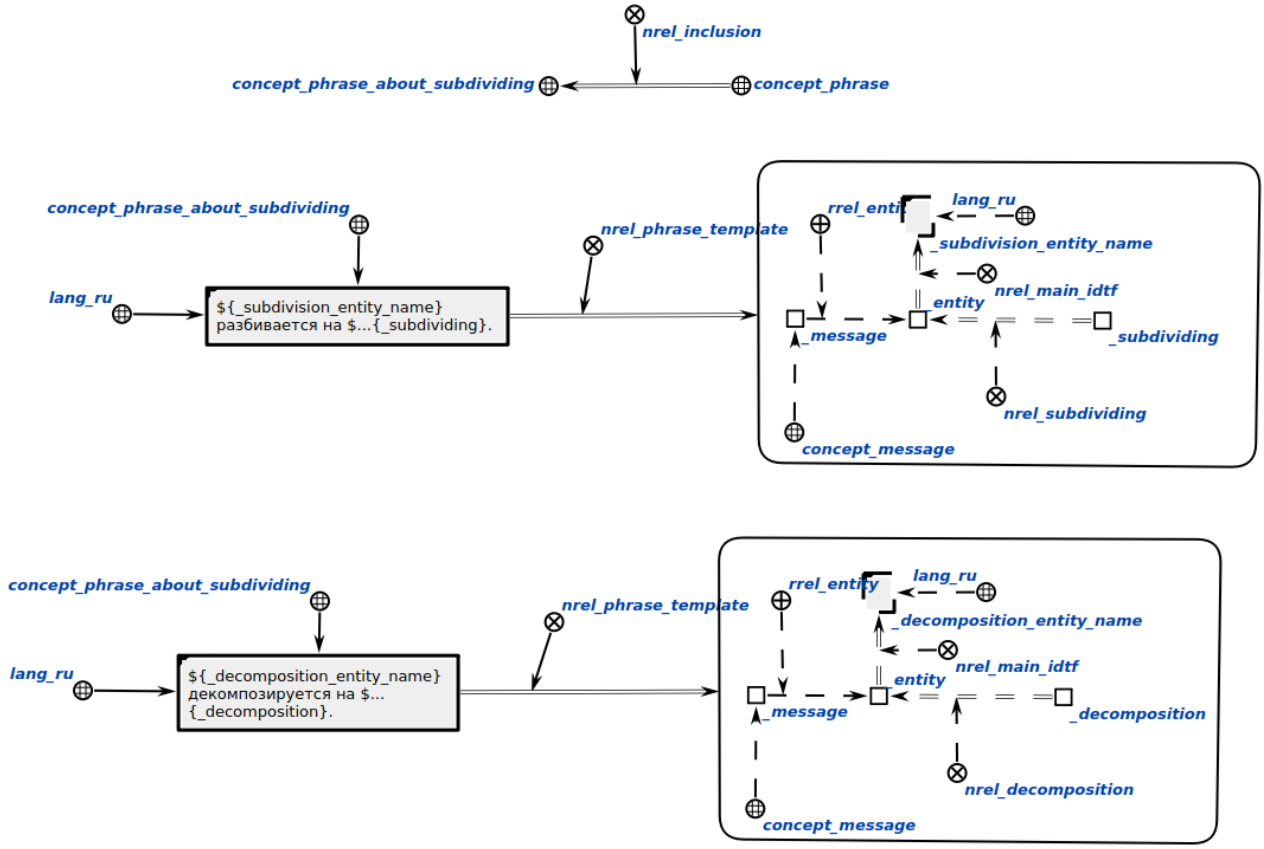

Рисунок А.11 – Пример формулы генерации фразы ответа

Список использованных источников

1 Голенков, В. В. Открытая технология онтологического проектирования, производства и эксплуатации семантически совместимых гибридных интеллектуальных компьютерных систем / В. В. Голенков, Н. А. Гулякина, Д. В. Шункевич. – Минск : Бестпринт, 2021. – 690 с.

2 Метасистема OSTIS [Электронный ресурс]. – Режим доступа: https://ims.ostis.net. – Дата доступа: 24.02.2025.

3 Гаврилова, Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. – СПб. : Питер, 2000. – 384 с.

4 Гаврилова, Т. А. Инженерия знаний. Модели и методы : учеб. / Т. А. Гаврилова, Д. В. Кудрявцев, Д. И. Муромцев. – СПб. : Лань, 2016. – 324 с.

5 MindMeister [Электронный ресурс]. – Режим доступа: https://www.mindmeister.com/ru. – Дата доступа: 25.01.2025.

6 Miro [Электронный ресурс]. – Режим доступа: https://miro.com/ru/. – Дата доступа: 25.01.2025.

7 Краткая презентация о Технологии OSTIS [Электронный ресурс]. – Режим доступа: https://docs.google.com/presentation/d/11j0c3FaolS3_gC3gsb4W_B18nAzC6d9RFhn_ZbvZGPU/edit?slide=id.p1#slide=id.p1. – Дата доступа: 03.04.2025.

8 Документация платформы, на которой спроектирована NIKA [Электронный ресурс]. – Режим доступа: https://ostis-apps.github.io/nika/. – Дата доступа: 03.04.2025.

9 Документация программного варианта реализации платформы интерпретации sc-моделей компьютерных систем [Электронный ресурс]. – Режим доступа: https://github.com/ostis-ai/ostis-web-platform/blob/develop/docs/main.pdf. – Дата доступа: 03.04.2025.

10 Презентация «Текущее состояние и направления развития технологий искусственного интеллекта и традиционных информационных технологий» [Электронный ресурс]. – Режим доступа: https://docs.google.com/presentation/d/1Uvl9C2Suinm6zCBdIEqtF2oKGrkpe1V_nEOH2bO-ey4/edit?slide=id.p1#slide=id.p1. – Дата доступа: 03.04.2025

11 Wit.ai [Electronic resource]. – Mode of access: https://wit.ai/. – Date of access: 03.04.2025.

12 Проект системы NIKA [Electronic resource]. – Mode of access: https://github.com/ostis-apps/nika. – Date of access: 03.04.2025.

13 Документация web-платформы OSTIS [Electronic resource]. – Mode of access: https://github.com/ostis-ai/ostis-web-platform. – Date of access: 03.04.2025.

14 Документация sc-машины OSTIS [Electronic resource]. – Mode of access: https://github.com/ostis-ai/sc-machine. – Date of access: 03.04.2025.

15 Редактор Linux KBE [Electronic resource]. – Mode of access: https://github.com/ostis-dev/kbe/releases/download/v0.4.0/KBE_4_0.tar.gz. – Date of access: 03.04.2025.

16 Редактор Windows KBE [Electronic resource]. – Mode of access: https://github.com/ostis-dev/kbe/releases/download/v0.4.0/KBE.4.0.ZIP. – Date of access: 03.04.2025.

17 Редактор KBE [Electronic resource]. – Mode of access: https://github.com/ostis-ai/kbe. – Date of access: 03.04.2025.

18 Документация агента классификации NIKA [Electronic resource]. – Mode of access: https://github.com/ostis-apps/nika/blob/main/docs/agents/messageTopicClassificationAgent.md. – Date of access: 03.04.2025.

19 Исходный код агента классификации Electronic resource]. – Mode of access: https://github.com/ostis-apps/nika/tree/main/problem-solver/cxx/messageClassificationModule. – Date of access: 03.04.2025.

20 Программа обработки [Electronic resource]. – Mode of access: https://github.com/ostis-apps/nika/tree/develop/kb/section_subject_domain_of_dialogues/section_subject_domain_of_dialog_control/examples. – Date of access: 03.04.2025.

21 Документация агентов программы обработки системы NIKA [Electronic resource]. – Mode of access: https://github.com/ostis-apps/nika/tree/main/docs/agents. – Date of access: 03.04.2025.

22 Клиент на Python [Electronic resource]. – Mode of access: https://github.com/ostis-ai/py-sc-client/blob/main/README.md. – Date of access: 03.04.2025.

23 Машина обработки знаний на Python [Electronic resource]. – Mode of access: https://github.com/ostis-ai/py-sc-kpm. – Date of access: 03.04.2025.

24 Клиент на TypeScript [Electronic resource]. – Mode of access: https://github.com/ostis-ai/ts-sc-client/blob/main/README.md. – Date of access: 03.04.2025.

25 Пример разработки агента на языке Python [Electronic resource]. – Mode of access: https://github.com/ostis-apps/nika/tree/develop/problem-solver/py. – Date of access: 03.04.2025.

Список рекомендованной литературы

1 Андрейчиков, А. В. Интеллектуальные информационные системы : учеб. / А. В. Андрейчиков, О. Н. Андрейчикова. – М. : Финансы и статистика, 2006. – 422 с.

2 Андрейчиков, А. В. Интеллектуальные информационные системы и методы искусственного интеллекта : учеб. / А. В. Андрейчиков, О. Н. Андрейчикова. – М. : ИНФРА-М, 2025. – 530 с.

3 Арсеньев, Ю. Н. Принятие решений : интегрированные интеллектуальные системы : учеб. пособие / Ю. Н. Арсеньев, С. И. Шелобаев, Т. Ю. Давыдова. – М. : ЮНИТИ-ДАНА, 2003. – 270 с.

4 Бабкин, Э. А. Методы представления знаний и алгоритмы поиска в задачах искусственного интеллекта : учеб. пособие / Э. А. Бабкин, Э. А. Козырев, И. В. Куркина. – Н. Новгород : НФ ГУ ВШЭ, 2005. – 146 с.

5 Гаврилова, Т. А. Интеллектуальные технологии в менеджменте : инструменты и системы : учеб. пособие / Т. А. Гаврилова, Д. И. Муромцев. – СПб. : Высш. шк. менеджмента, 2007. – 488 с.

6 Гаврилова, Т. А. Онтологический подход к управлению знаниями при разработке корпоративных информационных систем / Т. А. Гаврилова // Новости искусственного интеллекта. – 2003. – № 2.

7 Голенков, В. В. Модели представления и переработки знаний : метод. указания / В. В. Голенков, Н. А. Гулякина. – Минск : БГУИР, 1998. – 32 с.

8 Городецкий, В. И. Современное состояние технологии извлечения знаний из баз знаний и хранилищ данных / В. И. Городецкий, В. В. Самойлов, А. О. Малов // Новости искусственного интеллекта. – 2002. – № 3.

9 Джонс, М. Т. Программирование искусственного интеллекта в приложениях / М. Т. Джонс. – М. : ДМК Пресс, 2006. – 312 с.

10 Жданов, А. А. Автономный искусственный интеллект / А. А. Жданов. – М. : БИНОМ. Лаборатория знаний, 2008. – 359 с.

11 Змитрович, А. И. Интеллектуальные информационные системы : учеб. пособие / А. И. Змитрович. – Минск : ТетраСистемс, 1997. – 368 с.

12 Интеллектуальные информационные системы и технологии : учеб. пособие / Ю. Ю. Громов [и др.]. – Тамбов : ТГТУ, 2013. – 244 с.

13 Искусственный интеллект. В 3-х кн. Кн. 3: Программные и аппаратные средства : справочник / Под ред. В. Н. Захарова, В. Ф. Хорошевского. – М. : Радио и связь, 1990. – 368 с.

14 Кудрявцев, Д. А. Разработка проблемно-ориентированных онтологий / Д. А. Кудрявцев // Тр. десятой Национальной конференции по искусственному интеллекту с междунар. участием. КИИ-2006, Обнинск, 25–28 сентября. – Т. 1. – М. : Физматлит, 2006. – URL: https://raai.robofob.ru/resurs/papers/cai/2006/doklad/Kudryavtsev.doc (дата обращения 25.11.2025)

15 Люгер, Д. Ф. Искусственный интеллект : стратегии и методы решения сложных проблем / Д. Ф. Люгер ; пер. с англ. – 4-е изд. – М. : Вильямс, 2005. – 864 с.

16 Нечеткие гибридные системы. Теория и практика / под ред. Н. Г. Ярушкиной. – М. : Физматлит, 2007. – 207 с.

17 Остроух, А. В. Интеллектуальные информационные системы и технологии : моногр. / А. В. Остроух, Н. Е. Суркова. – Красноярск : Научно-инновационный центр, 2015. – 370 с.

18 Представление и обработка знаний в графодинамических ассоциативных машинах : моногр. / В. В. Голенков [и др.] ; под ред. В. В. Голенкова. – М. : БГУИР, 2001. – 412 с.

19 Рассел, С. Искусственный интеллект: современный подход / С. Рассел, П. Норвиг ; пер. с англ. – 2-е изд. – М. : Вильямс, 2007. – 1408 с.

20 Рыбина, Г. В. Основы построения интеллектуальных систем : учеб. пособие / Г. В. Рыбина. – М. : Финансы и статистика, 2021. – 432 с.

21 Рыбина, Г. В. Проектирование систем основанных на знаниях : учеб. пособие / Г. В. Рыбина. – М. : МИФИ, 1997. – 102 с.

22 Семантическая модель сложноструктурированных баз данных и баз знаний : учеб. пособие / В. В. Голенков [и др.]. – Минск : БГУИР, 2004. – 262 с.

23 Статические и динамические экспертные системы / Э. В. Попов [и др.]. – М. : Финансы и статистика, 1996. – 318 с.

24 Стефанюк, В. Л. Локальная организация интеллектуальных систем / В. Л. Стефанюк. – М. : Физматлит, 2004. – 328 с.

25 Тарасов, В. Б. От многоагентных систем к интеллектуальным организациям / В. Б. Тарасов. – М. : Эдиториал УРСС, 2002. – 352 с.

26 Технология комплексной поддержки жизненного цикла семантически совместимых интеллектуальных компьютерных систем нового поколения / под общ. ред. В. В. Голенкова. – Минск : Бестпринт, 2023. – 1064 с.

27 Уотермен, Д. Руководство по экспертным системам / Д. Уотермен. – М. : Мир, 1989. – 388 с.

28 Частиков, А. П. Разработка экспертных систем. Среда CLIPS / А. П. Частиков, Т. А. Гаврилова, Д. Л. Белов. – СПб. : БХВ-Петербург, 2003. – 608 с.

Св. план 2025, поз. 5

Учебное издание

Гракова Наталья Викторовна
Садовский Михаил Ефимович
Жмырко Александра Владимировна
Зотов Никита Владимирович

ТЕХНОЛОГИИ И ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *А. Ю. Шурко*
Компьютерная правка, оригинал-макет *Е. Г. Бабичева*

Подписано в печать 15.06.2026. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 8,95. Уч.-изд. л. 9,2. Тираж 70 экз. Заказ 176.

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