

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

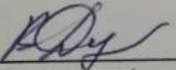
РЕАЛИЗАЦИЯ РАСПРЕДЕЛЕННОЙ ГРАФО-
ТРАНСФОРМАЦИОННОЙ СИСТЕМЫ НА ОСНОВЕ
ОНТОЛОГИЙ И ЯЗЫКА SPARQL

Студент

 30.05.24
(подпись, дата)

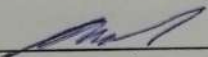
Ручкин Михаил Алексеевич
(ФИО полностью)

Руководитель

 30.05.24
(подпись, дата)

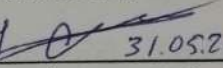
В.Н. Дубинин
(фамилия, инициалы)

Нормоконтролёр

 30.05.24
(подпись, дата)

А.С. Бычков
(фамилия, инициалы)

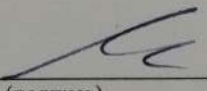
Рецензент

Доцент Пенз ГТУ  31.05.24
(должность, место работы) (подпись, дата)

Петерский С.В.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 № 11)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

06.06.2024 № 5)

Секретарь ГЭК


(подпись)

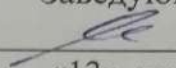
Е.И. Гурин
(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«13» сентября 2022 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Ручкин Михаил Алексеевич гр. 22ВВИм1 факультета вычислительной техники, направления 09.04.01

2. Тема работы «Реализация распределенной графо-трансформационной системы на основе онтологий и языка SPARQL»

Тема утверждена приказом ПГУ № 765/о от "13" сентября 2022 г.

3. Руководитель магистерской диссертации д.т.н., профессор Дубинин В.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки: иллюстрация графо-трансформационного подхода к проектированию сложных систем в среде семантического Web, а также созданию графодинамических моделей на основе технологической платформы семантического Веб.

Разработать методическое, лингвистическое, информационное и программное обеспечение для поддержки и реализации графо-трансформационного подхода к проектированию сложных систем в среде семантического Web. Показать применимость разработанных методов и средств на конкретных примерах.

5. Перечень подлежащих разработке вопросов:

1) обзор и анализ существующих работ по методам и средствам трансформации графов с использованием RDF-онтологий и языка SPARQL;

2) разработка онтологии типизированных атрибутивных графов;

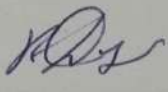
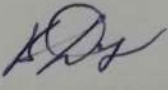
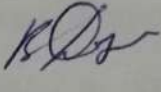
3) разработка методики реализации правил трансформации графов на языке SPARQL;

4) разработка языка описания моделей выполнения SPARQL UPDATE – правил;

5) разработка инструментальной программной системы для поддержки трансформации графов на основе RDF-онтологий и языка SPARQL;

6) реализация конкретных систем трансформации графов с использованием разработанного инструментария.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Обзор и анализ существующих работ по методам и средствам трансформации графов с использованием RDF-онтологий и языка SPARQL	15%	13.09.2022-16.12.2022	
2.	Разработка онтологии типизированных атрибутивных графов	10%	17.12.2022-18.02.2023	
3.	Разработка методики реализации правил трансформации графов на языке SPARQL	15%	19.02.2023-24.05.2023	
4.	Разработка языка описания моделей выполнения SPARQL UPDATE – правил	15%	25.05.2023-27.08.2023	
5.	Разработка инструментальной программной системы для поддержки трансформации графов на основе RDF-онтологий и языка SPARQL	20%	28.08.2023-31.12.2023	
6.	Реализация конкретных систем трансформации графов с использованием разработанного инструментария	15%	01.01.2024-04.04.2024	
7.	Написание пояснительной записки	10%	05.04.2024-27.05.2024	

Дата выдачи задания "13" сентября 2022 г.

Руководитель магистерской диссертации

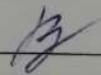


13.09.2022

Дубинин В.Н.

(подпись, дата)

Задание к исполнению принял студент



13.09.2022

Ручкин М.А.

(подпись, дата)

Работу к защите допустить
Декан факультета



03.06.2024

Фионова Л.Р.

(подпись, дата)

РЕФЕРАТ

Магистерская работа содержит 181 лист, 30 рисунков, 1 таблицу, 31 источник, 7 приложений

ОНТОЛОГИИ, СЕМАНТИЧЕСКИЙ ВЕБ, ТИПИЗИРОВАННЫЕ АТРИБУТНЫЕ ГРАФЫ, ТРАНСФОРМАЦИЯ ГРАФОВ, RDF, SPARQL.

Объектом исследования являются системы трансформации графов на основе онтологий и языка SPARQL.

Цель работы – разработка методического, лингвистического, информационного и программного обеспечения для поддержки и реализации графо-трансформационного подхода к проектированию сложных систем в среде семантического Веб.

В процессе работы выполнялся обзор и анализ существующих работ по методам и средствам трансформации графов с использованием RDF-онтологий и языка SPARQL.

В результате выполнения магистерской работы были разработаны: онтология типизированных атрибутивных графов, методика кодирования правил трансформации графов на языке SPARQL, язык описания моделей выполнения SPARQL Update – правил, инструментальная программная система для поддержки трансформации графов на основе RDF-онтологий и языка SPARQL и конкретные системы трансформации графов.

Результаты данной работы могут быть использованы в следующих областях: разработка информационных обрабатывающих систем и систем автоматизированного проектирования различного назначения в среде семантического Веб; учебный процесс по дисциплинам «Теории графов», «Базы данных и знаний», «Технологии разработки программного обеспечения», «Методы и моделирование», «Технологии семантического Веб» и др.

					ПГУ1.090401.20.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Реализация распределенной графо- трансформационной системы на основе онтологий и языка SPARQL	Лит.	Лист	Листов	
Разраб.		Ручкин М.А.		30.05.24					
Провер.		Дубинин В.Н.		30.05.24			2	181	
Н. Контр		Бычков А.С.		30.05.24		ФВТ, каф. ВТ			
Утверд.		Митрохин М.А.		30.05.24	Пояснительная записка	гр. 22ВВИМ1			

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ.	5
ВВЕДЕНИЕ.	6
1 Обзор и анализ методов и средств трансформации графов с использованием RDF-онтологий и языка SPARQL.	10
1.1 Краткий обзор литературы.	10
1.2 Методы трансформации графов.	13
1.3 Инструментальные средства трансформации графов.	14
1.4 Язык запросов SPARQL.	16
2 Разработка методик и моделей для реализации системы трансформации графов с использованием RDF-онтологий и языка SPARQL.	19
2.1 Онтология типизированных атрибутивных графов.	19
2.2 Методика кодирования правил трансформации графов в виде SPARQL Update – запросов.	23
2.3 Модели выполнения SPARQL Update – правил.	26
2.4 Язык SPARQL/C для задания пользовательских моделей выполнения.	31
3 Инструментальная программная системы для поддержки трансформации графов на основе языка SPARQL.	35
3.1 Выбранные технологии для разработки.	35
3.2 Назначение, особенности и функции программы.	35
3.3 Описание пользовательского интерфейса.	36
3.4 Алгоритм работы интерпретатора.	40
4 Примеры реализации конкретных систем трансформации графов с использованием разработанного инструментария.	43
4.1 Решение задачи поиска кратчайшего пути.	43
4.2 Вычисление суммы целых чисел на языке SPARQL.	47
4.3 Умножение матрицы на вектор на языке SPARQL.	51
ЗАКЛЮЧЕНИЕ.	57

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.	59
ПРИЛОЖЕНИЕ А Листинг программы.	63
ПРИЛОЖЕНИЕ Б Онтология задачи поиска кратчайших путей.	129
ПРИЛОЖЕНИЕ В SPARQL-запросы задачи поиска кратчайших путей.	146
ПРИЛОЖЕНИЕ Г Онтология вычисления суммы целых чисел.	154
ПРИЛОЖЕНИЕ Д Онтология умножения матрицы на вектор.	158
ПРИЛОЖЕНИЕ Е SPARQL-запросы умножения матрицы на вектор.	166
ПРИЛОЖЕНИЕ Ж Презентация.	169

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей магистерской диссертации применяются следующие сокращения и обозначения

ПТГ– правило трансформации графа

ТАГ– типизированный атрибутный граф

MDE (Model-driven engineering) – моделирования на основе моделей

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста

RDF (Resource Description Framework) – структура описания ресурсов

SPARQL (SPARQL Protocol and RDF Query Language) – язык запросов к данным, представленным по модели RDF, а также протокол для передачи этих запросов и ответов на них

ВВЕДЕНИЕ

Актуальность темы. Графы и их производные модели являются важным инструментом в области информатики и вычислительной техники, позволяя адекватно описывать структуру системы и проводить анализ и оптимизацию графовых моделей. Однако статичность графов ограничивает их применение в областях, где необходимо описание динамических процессов и изменений во времени. В последнее время концепция MDE стала широко распространенной, и основной идеей является преобразование моделей на всех этапах проектирования. Разработка новых методов преобразования графовых моделей может значительно способствовать этому направлению, поскольку способность к изменению графов заложена в системах перезаписи графов и графовых грамматиках. Эти математические модели могут быть основой для создания соответствующих программных систем.

В контексте моделирования на основе графовых структур и применения методов теории графов для анализа и оптимизации моделей, разработка новых методов преобразования графовых моделей может иметь значимое значение. Особенно это актуально в контексте семантического Веб, который остается важным направлением развития информационных технологий. Ключевые принципы семантического Веб, такие как использование структурированных данных и онтологий, могут вдохновить разработку новых методов трансформации графовых моделей, которые бы учитывали динамические изменения в данных и обеспечивали их эффективную обработку и анализ.

Также семантические технологии играют важную роль в различных областях, таких как поиск информации, разработка интеллектуальных агентов, интеграция данных и интернет вещей. Несмотря на то, что идеи семантического веба были предложены уже давно, их реализация продолжается, и в последние годы наблюдается рост интереса к этой области,

особенно в области развития искусственного интеллекта, машинного обучения и анализа больших данных.

С учетом вышесказанного, тема исследования, связанная с разработкой распределенной графо-трансформационной системы на основе онтологий и языка SPARQL, является актуальной.

Цели и задачи исследования. Целью работы является разработка методического, лингвистического, информационного и программного обеспечения для поддержки и реализации графо-трансформационного подхода к проектированию сложных систем в среде семантического Веб

Для достижения поставленной цели необходимо решить следующие основные задачи:

- анализ публикаций по теме трансформации графов на основе онтологий и языка SPARQL и существующих системы трансформации графов,
- разработка онтологии типизированных атрибутных графов,
- разработка методики кодирования правил трансформации графов на языке SPARQL,
- разработка языка описания моделей выполнения SPARQL Update – правил,
- разработка инструментальной программной системы для поддержки трансформации графов на основе RDF-онтологий и языка SPARQL,
- реализация конкретных систем трансформации графов с использованием разработанного инструментария.

Объектом исследования являются системы трансформации графов.

Предметом исследования являются методы реализации графов в виде RDF онтологий и SPARQL Update правил трансформации графов, а также их применение в графо-трансформационной системе.

Научная новизна. Научная новизна определяется следующими результатами:

– разработана онтология типизированных атрибутных графов (ТАГ), в основу которой положено их математическое определение. Особенностью онтологии является словарь, пригодный для описания любых ТАГ, что позволяет использовать онтологию при разработке автоматизированных систем, включающих обработку графов, на основе технологий семантического Веб,

– разработана методика кодирования правил трансформации графов в виде SPARQL Update запросов. Особенностью методики является использование паттерна DELETE-INSERT-SELECT-LIMIT, что позволяет одним запросом описывать одновременно и правую, и левую часть правила преобразования графа,

– предложены модели выполнения SPARQL Update – правил. Особенностью которых является возможность реализации задач сложной обработки данных, характерной для языков программирования высокого уровня,

– разработан язык описания пользовательских моделей выполнения SPARQL Update – правил, позволяющий разрабатывать произвольные процедуры обработки RDF – данных, что значительно повышает возможности языка SPARQL и позволяет построение информационных обрабатывающих систем в среде семантического Веб.

Практическая ценность работы. Практические результаты, полученные в результате исследований:

– разработана инструментальная программная система для поддержки трансформации графов на основе RDF-онтологий и языка SPARQL, позволяющая пользователям более гибко настраивать выполнение запросов и адаптировать их под конкретные задачи;

– разработаны онтология и правила преобразования для решения задачи поиска кратчайшего пути в графе, позволяющие оценить полезность использования онтологий и языка SPARQL в системах трансформации графов;

– показана применимость разработанной инструментальной программной системы для проведения числовых вычислений с использованием встроенного в эту систему языка описания моделей выполнения SPARQL Update – правил и собственно языка SPARQL. В качестве тестовых примеров рассматривались:

- 1) вычисление суммы целых чисел;
- 2) умножение матрицы на вектор.

1 Обзор и анализ методов и средств трансформации графов с использованием RDF-онтологий и языка SPARQL

1.1 Краткий обзор литературы

В работе [1] расширяют AToMPM, исследовательский мета-моделирующий и инструмент для переписывания графов, добавляя в него функциональность трансформации графов на основе RDF. Они решают проблему сопоставления графов, генерируя индивидуальные SPARQL-запросы для каждого шаблона правила, и используем это для эффективного выполнения сопоставлений на исходном графе RDF.

В статье [2] представлен язык преобразования для RDF, называемый STTL. Оригинальность и мощь STTL заключается в том, что он основан на SPARQL. Описывается общий движок правил преобразования, реализующий STTL, а также несколько преобразователей RDF, определенных для различных выходных языков.

В статье [3] представлен новый формальный фреймворк на основе теории категорий, который предоставляет четкие и краткие формальные определения основных особенностей RDF и SPARQL. Графы RDF, а также базовые графовые шаблоны SPARQL определяют, как объекты некоторых вложенных категорий. Рассматривается формализация операционной семантики базовых запросов SPARQL CONSTRUCT и SELECT с использованием нового подхода алгебраического преобразования графов, известного как POIM.

В работе [4] предлагается подход к объединению двух измерений анализа графов - онлайн аналитической обработки графов (OLGAP) и графового майнинга (GM). OLGAP сосредотачивается на сопоставлении подграфических шаблонов, в то время как GM автоматически извлекает знания из графов. Предложенный подход направлен на расширение функциональности тройных хранилищ RDF, которые изначально предназначены для хранения данных RDF и обеспечения возможностей OLGAP. Этот подход позволяет выполнять алгоритмы GM, такие как

PageRank и анализ связанных компонентов, на данных RDF, улучшая аналитические возможности систем RDF.

В работе [5] описывается потенциал использования алгебраических преобразований графов для обработки данных в формате RDF. Авторы отмечают, что несмотря на обширные исследования анализа RDF данных, еще не было достаточного внимания уделено концепции преобразования и манипулирования этими данными. Предложен подход, основанный на использовании алгебраических преобразований графов, которые обычно применяются для анализа и модификации графовых структур. Поскольку RDF данные могут быть представлены в виде графов, авторы считают, что алгебраические методы могут быть эффективно использованы для работы с ними. Рассмотрены особенности графов RDF, которые делают их интересными для применения алгебраических методов, и какие новые решения могут быть разработаны для таких задач, как атрибуция, типизация и управление уникальными узлами.

В работе [6] представлено исследование о том, как формальные методы алгебраического преобразования графов могут быть применены в контексте Семантического веба. Основной упор делается на том, как эти методы могут быть использованы для моделирования и формулирования операций преобразования данных на высоком уровне абстракции. Для демонстрации этой идеи, авторы разрабатывают небольшое исследование случая книжного магазина, чтобы показать, каким образом можно использовать предложенные методы для решения практических задач. В частности, они описывают, как операции модификации данных в этом контексте могут быть реализованы с помощью языка SPARQL/Update и подхода, основанного на правилах преобразования графов. Авторы подчеркивают, что основное преимущество их подхода к преобразованию на основе правил заключается в том, что он упрощает переход от разработки приложений к разработке правил, что в свою очередь способствует созданию общих решений и снижает сложность

обслуживания путем использования декларативной спецификации операций модификации данных.

В работе [7] предлагается новый способ преобразования онтологий RDF, основанный на языке запросов SPARQL. Он использует определенные шаблоны RDF, представляющие модули, которые могут быть взаимосвязаны. Основное внимание предложенного метода направлено на преобразования крупномасштабных наборов данных, где существующие методы требуют определений отображения, которые сложно поддерживать, то есть не эволюционирующие. Представлена краткая демонстрация, а также сравнение с другими языками преобразования.

В статье [8] предложен метод управления эволюцией баз данных RDF/S на основе правил трансформации графов в два этапа. В первом этапе правила формализуют изменения данных, обеспечивая согласованность графа. Во втором этапе обновления управляются с помощью генерации побочных эффектов, гарантируя соблюдение условий применения правил. Предложенный подход оптимизирует работу с побочными эффектами и обеспечивает гибкость в работе с недетерминизмом.

Статья [9] посвящена методу выполнения запросов к графовым данным, особенно атрибутивным графам, где вершины и рёбра имеют описательные атрибуты. Авторы предлагают новый параллельный движок выполнения запросов DG-SPARQL, который разработан для эффективной обработки запросов в таких графах. Они описывают дизайн и реализацию этого движка, его архитектуру и особенности, такие как распределение топологии графа по узлам и хранение данных графа в реляционном хранилище. Статья также включает анализ производительности DG-SPARQL и его сравнение с другими системами обработки графов, такими как Apache Giraph.

В статье [10] рассматривается методика реализации систем трансформации графов на языке SPARQL на примере модели задачи

взаимного исключения. однако в ней не предлагается общее правило/свод правил к кодированию правил трансформации графов.

В статье [11] рассматривается учебный пример модели книготорговой компании для изучения технологий RDF-хранилищ данных, включая язык запросов SPARQL.

В статье [12] предлагается онтологический подход к моделированию дискретных производственных систем. Для отображения динамики функционирования системы используется язык SPARQL. Подход иллюстрируется на примере тестовой и обрабатывающей станций производственной системы FESTO.

1.2 Методы трансформации графов

Существует несколько подходов к перезаписи графов, среди которых алгебраический подход, основанный на теории категорий. Алгебраический подход в свою очередь подразделяется на три разновидности: SPO (single-pushout), DPO (double-pushout) и pullback [13, 14]. В дальнейшем будем использовать SPO, поскольку это наиболее общий подход, и он реализован в одной из лучших систем трансформации графов AGG.

Пусть L , R , G и H будут помеченные графы. Прямая трансформация графа $G \Rightarrow t H$ определяется парой $t = (p, m)$, состоящей из правила замены графов $p: L \rightarrow R$ и инъективного графового морфизма (называемого соответствием) $m: L \sqsubset G$. Используя коммутативную диаграмму (pushout), представленную на рисунке 1, можно вычислить морфизмы $m': R \rightarrow H$ и $p': G \rightarrow H$. Последовательность $G_0 \Rightarrow G_1 \Rightarrow \dots \Rightarrow G_n$ прямых преобразований графа называется преобразованием графа и обозначается $G_0 \Rightarrow^* G_n$.

Отрицательное условие применения (NAC-условие) для правила p - это графовый морфизм $\text{nac}: L \rightarrow L^-$. Прямая трансформация графа $G \Rightarrow (p, m) H$ удовлетворяет NAC-условию, если не существует графовый морфизм $m^-: L^- \rightarrow G$ такой, что $m^- \cdot \text{nac} = m$. Упрощенно говоря, NAC-условие - это граф, который определяет запрещенную графовую структуру. К одному правилу

может быть присоединено несколько NAC-условий. В этом случае правило применимо, если удовлетворяются все NAC-условия.

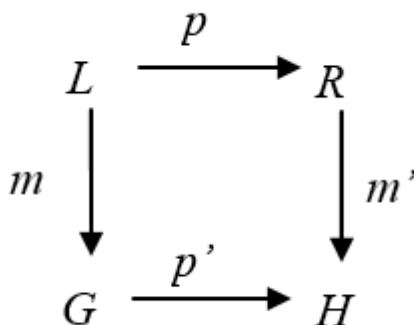


Рисунок 1 – Схематичное представление прямого вывода в виде универсального квадрата

Атрибутный граф – это граф, вершины и дуги которого помечены абстрактными типами данных. В случае атрибутных графов для применимости правила необходимо также выполнение условий по атрибутам вершин и дуг. При выполнении правила производится передача и вычисление атрибутов определенной части результирующего графа. Конфликт между трансформациями графов может быть найден путем определения критических пар правил замены графов.

Графом типов (type graph) называется некоторый фиксированный граф TG, представляющий информацию о типах узлов и дуг графов, генерируемых графовой грамматикой (ГГ). Назначение типов графу G с помощью графа TG задается полным графовым морфизмом $t: G \rightarrow TG$. Если существует графовый морфизм t для G, то граф G называется типизированным. Ограничения на структуру взаимосвязи экземпляров типов узлов могут задаваться при помощи кардинальности (множественности) роли связи. Кроме того, можно также задавать и множественность узлов.

1.3 Инструментальные средства трансформации графов

Способность к изменению графовых моделей заложена в системах перезаписи графов (graph rewriting system) и графовых грамматиках (graph grammar). На основе этих математических моделей могут быть созданы соответствующие программные системы. В настоящий момент к известным

программным системам трансформации графов относятся AGG [15] [16], GROOVE [17], GReAT [18], GrGen [19], AToM3 [20], QVT [21], ATL [22], VIATRA [23], Fujaba [24], PROGRES [25] и др. Рассмотрим более подробно шести наиболее популярным систем трансформации графов. Данные системы объединяет наличие у них графического интерфейса, поддержка выполнения на платформе Windows и бесплатное распространение.

AGG (Attributed Graph Grammar) — это система преобразования атрибутивных графов, поддерживающая алгебраический подход к преобразованию графов. AGG можно использовать как механизм преобразования графов общего назначения в приложениях JAVA, использующих методы преобразования графов. Подход к преобразованию графов в этой системе базируется на графовых грамматиках, а не на запрограммированных преобразованиях. Используя этот метод, преобразование начинается с начального входного графа, а заканчивается, когда не будет разрешенных правил преобразования. Но данный подход может оказаться нежизнеспособным, если пользователю потребуется явный контроль над последовательностью преобразований.

AToM3 (A Tool for Multi-formalism and Meta-Modelling) — это система для мультипарадигмального моделирования. Двумя основными задачами AToM3 являются мета-моделирование и преобразование модели. Мета-моделирование относится к описанию или моделированию различных видов формализмов, используемых для моделирования систем, возможности AToM3 на этом не ограничиваются. Преобразование модели относится к процессу преобразования, перевода или модификации модели в данном формализме в другую модель, которая может быть или не быть в том же формализме. В AToM3 формализмы и модели описываются как графы. Преобразования модели выполняются путем перезаписи графа. Таким образом, сами преобразования могут быть декларативно выражены в виде графо-грамматических моделей.

FUJABA (From UML to Java and back again) ориентирована на создание кода Java из моделей UML таким образом, чтобы модели UML можно было реконструировать из сгенерированного кода Java путем обратного преобразования. Fujaba использует метод, известный как моделирование на основе историй, в котором диаграмма действий указывает порядок, в котором должны применяться правила преобразования графа.

GReAT (The Graph Rewriting and Transformation) — это система для преобразования моделей с использованием методов преобразования графов. GReAT использует метамодели для указания абстрактного синтаксиса входных и целевых моделей (т.е. языков моделирования), а также правила последовательной перезаписи графа для определения самого преобразования.

PROGRES (PROgrammed Graph REwriting Systems) — еще одна система преобразования общего назначения. Обычно она используется для моделирования процессов прототипирования и инструментов реинжиниринга. PROGRES сочетает в себе UML-подобные диаграммы классов и правила перезаписи графов, которые работают с этими диаграммами. Обладает возможностью перевода готового преобразования в код Си для быстрого прототипирования.

GROOVE (Graphs for Object-Oriented Verification) основан на использовании простых графов для моделирования структуры объектно-ориентированных систем во время разработки, компиляции и выполнения, а также преобразований графов в качестве основы для преобразования модели. Это влечет за собой формальную основу для преобразования модели, а также возможность проверки преобразования модели посредством анализа результирующих систем преобразования графа, например, с использованием проверки модели.

1.4 Язык запросов SPARQL

Исходный граф, представленный в формате RDF, является ориентированным графом с маркированными ребрами, используемым для представления информации в сети Интернет. Преобразование этого графа

осуществляется с использованием языка запросов SPARQL, спецификация которого определяет синтаксис и семантику запросов для работы с данными в формате RDF [26].

Результатом выполнения запросов SPARQL может быть набор результатов или новый RDF-граф, в зависимости от поставленной задачи. Для успешной интерпретации и выполнения запросов SPARQL важно понимать его основные особенности, включая синтаксис запросов, правила формирования запросов и их семантику. Как и в SQL, SPARQL декларирует не только чтение, но и манипуляцию данными. Стандарт SPARQL Update описывает команды INSERT и DELETE для добавления и удаления триплетов в хранилище. Однако, обе эти команды оперируют той или иной выборкой, так что основа SPARQL — это команда SELECT.

Общая схема SPARQL-запроса SELECT выглядит так:

```
PREFIX
    # префиксные объявления - служат для указания сокращений
URI,
    # используемых в запросе.
FROM ...
    # источники запроса - определяют какие RDF-графы
запрашиваются.
SELECT ...
    # состав результата - определяет возвращаемые элементы
данных.
WHERE {...}
    # шаблон запроса - определяет, что запрашивать из набора
данных.
```

Ниже приведена часть используемых ключевых слов в SPARQL запросах:

- PREFIX — служит для сокращения URI,
- OPTIONAL — обозначает необязательный шаблон,
- GRAPH — применяет шаблон к именованным графам,
- DISTINCT — обеспечивает уникальность решений в ответе на запрос,
- LIMIT — задаёт максимальное количество выводимых результатов,
- OFFSET — опускает в результате первые n решений,

– ORDER BY — позволяет отсортировать результат по возрастанию (ASC) или по убыванию (DESC).

2 Разработка методик и моделей для реализации системы трансформации графов с использованием RDF-онтологий и языка SPARQL

2.1 Онтология типизированных атрибутных графов

В рамках данной работы была разработана RDF-онтология типизированных атрибутных графов. Описание данной онтологии ТАГ было опубликовано в [27]. С использованием ТАГ возможно описание структуры практически любых сложных систем. Часто с использованием ТАГ описывают диаграммы различных видов (например, UML, ERD, IDEF, DFD и т.д.).

Формальное описание ТАГ в зарубежных источниках дается с алгебраической точки зрения (например, [28]). Ниже предлагается формальное описание ТАГ, более подходящее для их онтологического представления.

Типизированный атрибутный граф определяется следующим кортежем:

$$G = (N, E, NT, ET, AT, A, D, tn, te, tan, tae, at, na, ea, dom, val),$$

где N – множество вершин (узлов) графа,

$E \subseteq N \times N$ – множество направленных ребер (дуг) графа,

NT – множество типов узлов,

ET – множество типов дуг,

AT – множество типов (иначе, имен) атрибутов,

A – множество (экземпляров) атрибутов,

D – множество доменов (областей значений данных), $D = \{D_1, D_2, \dots, D_f\}$. В частном случае, это множество может быть представлено стандартными именами типов данных, например, $D = \{Boolean, Integer, String\}$,

$tn: N \rightarrow NT$ – сюръективное отображение, назначающее узлам соответствующие типы,

$te: N \rightarrow ET$ – сюръективное отображение, назначающее дугам соответствующие типы,

$tan: NT \rightarrow 2^{AT}$ – функция, назначающая типам узлов множества типов атрибутов,

$tae: ET \rightarrow 2^{AT}$ – функция, назначающая типам дуг множества типов атрибутов,

$at: A \rightarrow AT$ – сюръективное отображение, назначающее атрибутам соответствующие типы. Данное отображение может использоваться для разбиения множества атрибутов на классы, каждый из которых соответствует своему типу: $A = \{A_1, A_2, \dots, A_h\}$,

$na: N \rightarrow 2^A$ – функция, назначающая узлам множества атрибутов,

$ea: E \rightarrow 2^A$ – функция, назначающая дугам множества атрибутов,

$dom: AT \rightarrow D$ – функция, определяющая область значений типов атрибутов,

$val: A \rightarrow D_1 \cup D_2 \cup \dots \cup D_f$ – функция значений атрибутов.

Введем отношения эквивалентности @ и #:

$$n_i @ n_j \Leftrightarrow tn(n_i) = tn(n_j)$$

$$e_i \# e_j \Leftrightarrow te(e_i) = te(e_j)$$

Используя понятие фактор-множества произведем деление множества узлов и дуг на классы (типы):

$$N/@ = \{N_1, N_2, \dots, N_p\}$$

$$E/\# = \{E_1, E_2, \dots, E_q\}$$

$\varphi: 2^A \rightarrow 2^{AT}$ – функция, преобразующая множества атрибутов в соответствующие множества их типов. Данная функция является производной и строится на основе функции at .

Должно существовать взаимно однозначное соответствие между множеством типов атрибутов, назначенных узлу, и множеством типов атрибутов, назначенных типу этого узла. Подобное требование должно выполняться и по отношению к дугам:

$$\forall x \in N [\varphi(na(x)) \leftrightarrow \neg \varphi(tn(x))] \quad (1)$$

$$\forall x \in E [\varphi(ea(x)) \leftrightarrow \neg \varphi(te(x))]$$

Значение атрибута должно соответствовать области его допустимых значений, что определяется следующим ограничением:

$$\forall x \in A [val(x) \in dom(at(x))]$$

На основе предлагаемого формального описания ТАГ была построена ТВох-онтология ТАГ, представленная на рисунке 2. В этой онтологии определен словарь, пригодный для описания любых ТАГ. Классы онтологии представлены в виде овалов, в которые вписаны имена классов, типы данных – в виде прямоугольников со скругленными краями, а свойства – в виде дуг. Следует отметить, что в свойстве `has_attr_datatype` типы данных (`Boolean`, `Integer`, `String`) используется только декларативно в свойстве `rdfs:range`, без использования конкретных значений. Иными словами, триплетов, содержащих `has_attr_datatype` в качестве предиката, не существует. В таблице 1 приведено соответствие математических объектов элементам онтологии.

Для проверки консистентности описания ТАГ используются аксиомы классов. Например, аксиома, утверждающая, что каждая дуга должна иметь одну вершину-источник дуги и одну вершину-приемник дуги, может быть определена так:

Edge and (comes_to exactly 1 Node and comes_from exactly 1 Node)

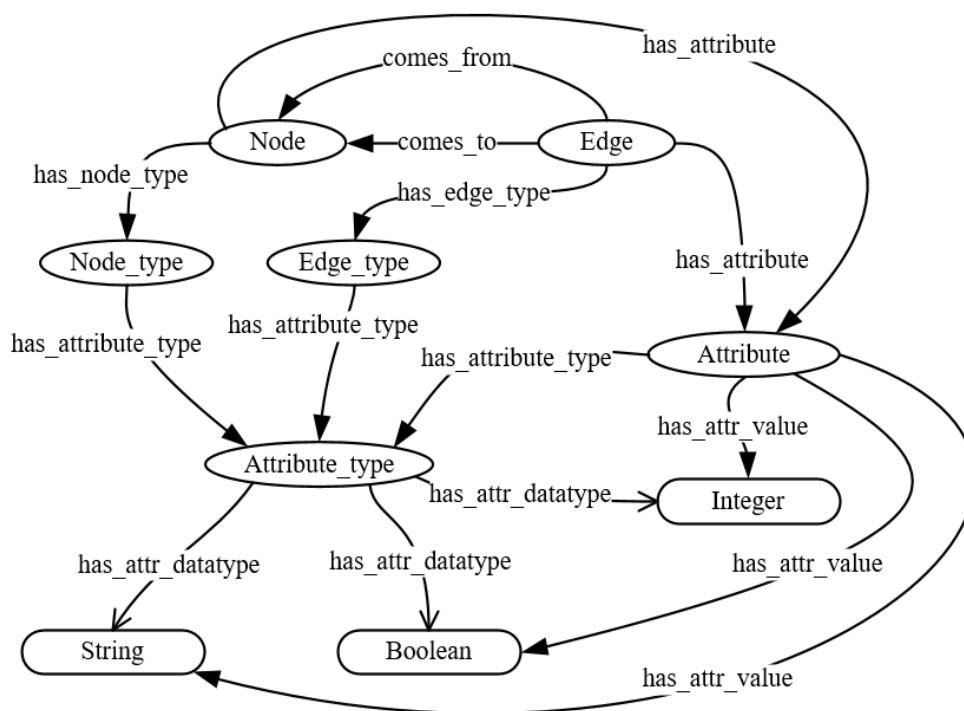


Рисунок 2 – Визуальное представление TBox - онтологии ТАГ

Таблица 1 – Соответствие математических объектов элементам онтологии

Математический объект	Элемент онтологии
Множество N	Класс Node
Множество E	Класс Edge
Множество NT	Класс Node_type
Множество ET	Класс Edge_type
Множество A	Класс Attribute
Множество AT	Класс Attribute_type
Сюръективное отображение tn	Объектное свойство <code>has_node_type</code>
Сюръективное отображение te	Объектное свойство <code>has_edge_type</code>
Функции tan и tae	Объектное свойство <code>has_attribute_type</code>
Сюръективное отображение at	Объектное свойство <code>has_attribute_type</code>
Функции na и ea	Объектное свойство <code>has_attribute</code>
Функция dom	Свойство по данным <code>has_attr_datatype</code>
Функция val	Свойство по данным <code>has_attr_value</code>

Наличие (вернее, отсутствие) взаимно однозначного соответствия, представленного формулой (1), может быть проверено с помощью следующего *SWRL*-правила:

```
Node(?x) ^ has_node_type(?x, ?y) ^ Node_type(?y) ^  
has_attribute_type (?y, ?k) ^ Attribute_type(?k) ^  
has_attribute(?x, ?n) ^ Attribute(?n) ^ has_attribute_type (?n,  
?m ) ^ Attribute_type(?m) ^ differentFrom(?k, ?m) ->  
NodeWithError(?x)
```

Онтологическое представление ТАГ дает возможность исследовать и выполнять операции над графами при помощи *SPARQL*-запросов. В качестве примера ниже представлен запрос для поиска достижимых узлов из конкретного узла *B* (с учетом циклов):

```
SELECT ?x WHERE { :B ( ^:comes_from/:comes_to)+ ?x }
```

2.2 Методика кодирования правил трансформации графов в виде SPARQL Update - запросов

Рассмотрим подход к реализации кодирования ПТГ в виде *SPARQL Update* – запросов. При описании ПТГ, обычно используется подход, где правило состоит из двух частей: левой и правой.

Левая часть ПТГ играет важную роль в определении условий применения правила. В этой части описываются конкретные условия, которые должны быть выполнены, чтобы правило могло быть активировано. Эти условия определяют, какие узлы, связи или свойства графа будут подвергнуты изменениям. Также в левой части указывается формальная структура графа, то есть конкретные элементы графа или их комбинации, которые являются объектами трансформации. Это могут быть как отдельные узлы и связи, так и их совокупности, удовлетворяющие определенным критериям.

Правая часть ПТГ, с другой стороны, определяет изменения, которые должны произойти с элементами графа, соответствующими условиям из левой части. Здесь также указывается формальная структура графа, описывающая, какие новые узлы, связи или свойства должны быть добавлены, изменены или удалены в результате трансформации.

Такая структура правила позволяет формально описывать, какие изменения должны произойти в графе в зависимости от определенных условий.

Далее будет рассмотрена предлагаемая общая методика кодирования ПТГ.

Левая часть ПТГ вместе с NAC- и AC-условиями представляется в WHERE-предложении с вложенным подзапросом, а именно в виде следующей конструкции:

```
WHERE {  
  SELECT <список параметров>  
  WHERE {<граф левой части ПТГ>, <NAC-условия>, <AC-условия>}  
  LIMIT 1}
```

При этом <список параметров> есть список переменных, представляющих вершины и дуги графа (а также их атрибуты) из левой части ПТГ, участвующих в правой части ПТГ и, соответственно в DELETE- и INSERT-предложениях SPARQL Update-запросов.

Правая часть ПТГ представляется с помощью DELETE- и INSERT-предложений SPARQL Update-запросов. Предложение DELETE используется для удаления вершин и дуг графа, а предложение INSERT – для их создания. В совокупности конструкция DELETE- INSERT используется для модификации значений атрибутов вершин и дуг графа.

Если одна и та же вершина имеется как левой части ПТГ, так и в правой его части, то она представляется одной и той же переменной языка SPARQL. Создание такой вершины не производится в SPARQL Update-запросе.

Если одна и та же дуга имеется как левой части ПТГ, так и в правой его части, то она представляется одной и той же переменной языка SPARQL. Создание такой дуги не производится в SPARQL Update-запросе.

Если некоторая вершина имеется в правой части ПТГ, а в левой ее нет, то производится ее создание с помощью INSERT-предложения с использованием пустого узла.

Если некоторая дуга имеется в правой части ПТГ, а в левой ее нет, то производится ее создание с помощью INSERT-предложения с использованием пустого узла. Связывание вновь созданного индивида для дуги с окружением производится с использованием объектных свойств comes_from и comes_to.

Если в правиле ПТГ имеется NAC-условие, то оно представляется FILTER NOT EXISTS – предложением.

Если в правиле ПТГ имеется несколько NAC-условий, то они представляются конструкцией

```
FILTER NOT EXISTS
    {<графовый шаблон для запрещенного графа 1>}
FILTER NOT EXISTS
    {<графовый шаблон для запрещенного графа 2>}
...
FILTER NOT EXISTS
    {<графовый шаблон для запрещенного графа N>}
```

Арифметические выражения, определяющие изменение значений атрибутов вершин и дуг, представляются при помощи BIND-предложения.

Если в правиле ПТГ имеется AC-условие, то оно напрямую представляется FILTER–предложением.

При формировании шаблона графа левой части надо указать, что все участвующие в нем вершины графа попарно различны. Это можно сделать при помощи предложений FILTER.

Если принята послойная (layered) дисциплина выполнения правил, то правило из слоя выполняется до тех пор, пока оно не станет неразрешенным. Слои правил выполняются в порядке увеличения их порядкового номера.

Надо обеспечить явно, что все вершины графа левой части правила являются попарно различными. Для этого используется конструкция FILTER, например, если используются три вершины, то нужно написать:

```
FILTER (?c1 != ?c2)
FILTER (?c1 != ?c3)
FILTER (?c2 != ?c3)
```

Если некоторая вершина имеется в левой части ПТГ, а в правой ее нет, то производится ее удаление с помощью DELETE-предложения. При этом

нужно учитывать, если вершина имеет связи с любыми другими вершинами, то необходимо удалить все связи перед удалением самой вершины.

Если некоторая дуга имеется в левой части ПТГ, а в правой ее нет, то производится ее удаление с помощью DELETE-предложения. Перед удалением индивида дуги удаляются его связи с окружением использующие объектные свойства `comes_from` и `comes_to`.

2.3 Модели выполнения SPARQL Update – правил

Управление выполнением SPARQL Update-правил играет ключевую роль в обеспечении эффективной и надежной обработки операций преобразований RDF-онтологий. Для решения данной задачи были разработаны модели выполнения, представленные далее.

Для (автоматического) управления выполнением SPARQL Update-правил могут использоваться два класса моделей:

- предопределенные элементарные модели выполнения,
- модели выполнения, определяемые пользователем.

Для реализации моделей выполнения используются два основных типа запросов:

- SPARQL Update-запрос, имеющий структуру DELETE-INSERT-WHERE. По сути дела, это специализированное продукционное правило ЕСЛИ <Условие>, ТО <Действие>. Условная часть правила определяется WHERE-предложением, а действия определяются предложениями DELETE и INSERT, позволяющими добавлять и удалять RDF-триплеты,
- ASK-запрос, содержащий в своей структуре только <Условие> (предложение WHERE). Упрощенно говоря, данный запрос позволяет определить истинность или ложность условия.

В основе определения всех моделей лежат два действия:

- выполнение (вернее, запуск) SPARQL Update-правила, причем правило может как сработать, так и не сработать. Правило срабатывает при истинной условной части правила, и, соответственно, не срабатывает при ложной. Будем обозначать такое действие как `exes`,

– определение разрешенности правила, что осуществляется с помощью ASK-запроса. Будем обозначать такое действие как А.

SPARQL Update-правило называется разрешенным, если удовлетворяется условие, определяемое предложением WHERE. Это происходит в том случае, когда существует такая подстановка переменных, что условие становится истинным. Соответственно, SPARQL Update-правило является не разрешенным, если не существует такой подстановки переменных, при котором условие будет истинным.

Предлагается следующая классификация (предопределенных) элементарных моделей выполнения:

- упорядоченные модели выполнения,
- приоритетная модель выполнения,
- рандомная модель выполнения.

В упорядоченных моделях выполнения в множестве правил задается линейный порядок, который используется при их выполнении. В дальнейшем будет использоваться алгоритмическое представление моделей выполнения в виде блок-схем или на псевдокоде, которое, по сути, определяет операционную семантику этих моделей.

Для классификации упорядоченных моделей выполнения используются два классификационных признака:

- P1: Циклическое выполнение правила / Ациклическое выполнение правила [C/A],
- P2: Однократное / Многократное выполнение группы правил [S/M].

Таким образом, упорядоченная модель выполнения может идентифицироваться (кодироваться) следующим образом:

$$O\langle P1\rangle\langle P2\rangle,$$

где О – это признак упорядоченной модели.

В алгоритмах будут использоваться следующие обозначения:

- $\text{exec } R_i$ – выполнение (запуск) SPARQL Update-правила R_i ,
- $A(R_i)$ – условие, является ли SPARQL Update-правило R_i разрешенным,
- $\text{orAll} = A(R_1) \vee A(R_2) \vee \dots \vee A(R_n)$ - условие, является ли хотя бы одно SPARQL Update-правило из определенного набора разрешенным.

2.3.1 Ациклическая однопроходная модель выполнения (OAS)

Краткая характеристика: Правила выполняются последовательно и однократно, одно за другим, за один проход.

Спецификация: $\text{OAS}(R_1, R_2, \dots, R_n)$.

Интерпретация на псевдокоде:

```

exec R1;
exec R2;
...
exec Rn.

```

2.3.2 Ациклическая многопроходная модель выполнения (OAM)

Краткая характеристика: Правила выполняются последовательно и однократно, одно за другим, за несколько проходов, до тех пор, пока все правила не окажутся не разрешенными.

Спецификация: $\text{OAM}(R_1, R_2, \dots, R_n)$.

Интерпретация на псевдокоде:

```

repeat
    exec R1;
    exec R2;
    ...
    exec Rn
until not (A(R1) ∨ A(R2) ∨ ... ∨ A(Rn)) .

```

2.3.3 Циклическая однопроходная модель выполнения (OCS)

Краткая характеристика: Правила выполняются последовательно многократно, одно за другим, за один проход. Каждое правило выполняется до тех пор, пока оно является разрешенным.

Следует отметить, что при использовании этой модели существует риск закливания выполнения в том случае, если выполнение правила никогда не приводит к его неразрешенности. В этом случае надо проводить

контроль выполнения (например, с помощью счетчика циклов) для исключения зацикливаний.

Спецификация: $OCS(R_1, R_2, \dots, R_n)$.

Интерпретация на псевдокоде:

```
while A(R1) exec R1;  
while A(R1) exec R2;  
...  
while A(R1) exec Rn.
```

2.3.4 Циклическая многопроходная модель выполнения (ОСМ)

Краткая характеристика: Правила в наборе выполняются последовательно циклически, одно за другим, за несколько проходов, до тех пор, пока все правила не окажутся не разрешенными. В рамках данной модели в отдельном проходе каждое правило выполняется до тех пор, пока оно является разрешенным.

Спецификация: $OCM(R_1, R_2, \dots, R_n)$.

Интерпретация на псевдокоде:

```
repeat  
    while A(R1) exec R1;  
    while A(R1) exec R2;  
    ...  
    while A(R1) exec Rn;  
until not (A(R1)  $\vee$  A(R2)  $\vee$  ...  $\vee$  A(Rn)) .
```

2.3.5 Приоритетная модель выполнения (PRIOR)

Краткая характеристика: Правила в наборе выполняются в соответствии с приоритетом, который определяется положением правила в наборе (то есть, линейным порядком) – чем ближе к началу списка, тем приоритетнее правило. В отличие от упорядоченных моделей выполнения приоритет правила поставлен во главу угла. При этом при любом раскладе первым будет выполняться самое приоритетное правило. Наиболее близкой является модель № 4, но там это не гарантируется, поскольку выполнение одного правила может к активации или деактивации другого правила. Например, выполнение второго правила может привести к активизации

первого (самого приоритетного) правила, но это первое правило будет выполнено только после полного окончания текущего прогона.

Спецификация: $\text{PRIOR}(R_1, R_2, \dots, R_n)$.

Интерпретация на псевдокоде:

```
repeat
  repeat
    repeat
      exec R1
    until not (A(R1))
    exec R2
  until not (A(R1)  $\vee$  A(R2))
  ...
  exec Rn
until not (A(R1)  $\vee$  A(R2)  $\vee$  ...  $\vee$  A(Rn)) .
```

2.3.6 Рандомная модель выполнения (RAND)

Краткая характеристика: Правило для выполнения выбирается случайным образом. Выполнение правил заканчивается только тогда, когда не остается ни одного разрешенного правила.

Rand – функция вычисления случайного числа в диапазоне от 1 до n.

C – условный оператор case на несколько выходов, осуществляющий анализ сгенерированного случайного числа и выбор соответствующей выходной ветки.

Спецификация: $\text{RAND}(R_1, R_2, \dots, R_n)$.

Интерпретация на псевдокоде:

```
repeat
  c=rand(1,n)
  case c
  1: exec R1;
  2: exec R2;
  ...
  n: exec Rn;
until not (A(R1)  $\vee$  A(R2)  $\vee$  ...  $\vee$  A(Rn)) .
```

2.3.7 Пользовательские модели выполнения на языке SPARQL/C

Если ни одна из предопределенных элементарных моделей (или их композиции) не подходит для решения поставленной задачи, то пользователь может сформировать свою модель выполнения правил с использованием

собственного простого языка SPARQL/C интерпретирующего типа, включающего следующие операторы:

- составной оператор do-end,
- условный оператор if-then-else,
- оператор цикла repeat-until,
- оператор цикла while.

Данные операторы являются традиционными для большинства языков программирования высокого уровня, их семантика хорошо известна и не требует подробных объяснений. Особенностью данных операторов в контексте моделей выполнения правил является использование в них трех действий (частично описанных выше):

- $exes R_i$ – запуск i -го SPARQL-UPDATE-запроса,
- $A(R_i)$ – определение того, есть ли решения для i -го SPARQL-UPDATE-запроса (реализуется ASK-запросом),
- $C(R_i)$ – число найденных решений для i -го SPARQL-UPDATE-запроса (реализуется COUNT-запросом).

Следует отметить, что все предопределенные элементарные модели выполнения можно представить на языке SPARQL/C. Данные модели можно определить, как некие макросы (макро-действия) на языке SPARQL/C:

- OAS – ациклическая однократная модель выполнения,
- OAM – ациклическая многократная модель выполнения,
- OCS – циклическая однократная модель выполнения,
- OCM – циклическая многократная модель выполнения,
- PRIOR – приоритетная модель выполнения,
- RAND – случайная модель выполнения.

2.4 Язык SPARQL/C для задания пользовательских моделей выполнения

Язык SPARQL/C предназначен для задания пользовательских моделей выполнения SPARQL UPDATE – правил. При этом модели выполнения

представляются в виде программ специального вида (скриптов), которые интерпретируются в процессе их выполнения. Данный язык был разработан совместно с научным руководителем.

Ниже представлен синтаксис (грамматика) языка SPARQL/C с использованием расширенных форм Бэкуса-Наура (РБНФ). Данный синтаксис является «минималистским», без всякого «синтаксического сахара». В нем сведено к минимуму использование скобок, точек с запятой и других разделителей.

```
Последовательность_операторов ::= Оператор {Оператор}
```

```
Оператор ::=
```

```
    Оператор_выполнения_предопределенной_модели  
        | Оператор_запуска_правила  
        | Составной_оператор  
        | Условный_оператор  
        | Оператор_цикла_while  
        | Оператор_цикла_repeat_until
```

```
Оператор_выполнения_предопределенной_модели ::=
```

```
    Имя_предопределенной_модели "(" Список_правил ")"
```

```
Список_правил ::= Идент_правила {"," Идент_правила}
```

```
Имя_предопределенной_модели ::=
```

```
    "OAS" | "OAM" | "OCS" | "OCM" | "PRIOR" | "RAND"
```

```
Оператор_запуска_правила ::= "exec" Идент_правила
```

```
Идент_правила ::= Идентификатор
```

```
Составной_оператор ::=
```

```
    "do" Последовательность_операторов "end"
```

```
Условный_оператор ::=
```

```
    "if" Лог_выражение "then" Оператор  
    [ "else" Оператор]
```

```
Оператор_цикла_while ::=
```

```
    "while" Лог_выражение  
    Оператор
```

```
Оператор_цикла_repeat_until ::=
```

```
    "repeat"  
    Последовательность_операторов  
    "until" Лог_выражение
```

```

Лог_выражение ::= Лог_терм {"or" Лог_терм}

Лог_терм ::= Лог_множитель {"and" Лог_множитель}

Лог_множитель ::=
    Разрешенность_правила
    | "not" Лог_выражение
    | "(" Лог_выражение ")"
    | Число_решений_правила Операция_сравнения Целое

Операция_сравнения ::= "=" | "<>" | "<" | "<=" | ">" | ">="

Разрешенность_правила ::= "A" ("Идент_правила")
Число_решений_правила ::= "C" ("Идент_правила")

```

Язык SPARQL/C (кроме узкой задачи реализации моделей выполнения правил) может также использоваться для реализации задач сложной обработки данных, характерной для языков программирования высокого уровня (например, суммирование элементов массива, поиск максимума и минимума в массиве, операции с массивами и строками, обработка сложных структур данных и т.д.).

Ниже приведены простейшие примеры программ (скриптов) на языке SPARQL/C.

Пример 1:

```
if A(R1) then exec R1 else exec R2
```

Пример 2

```

if A(R1) then
    do
        exec R1
        exec R2
    end
else
    do
        exec R3
        exec R4
    end

```

Пример 3 (эквивалентен примеру 2):

```
if A(R1) then OAS(R1,R2) else OAS(R3,R4)
```

Пример 4:

```

while A(R1) or A(R2)
do

```

```

        exec R1
        exec R2
    end

```

Пример 5:

```

repeat
    if A(R1) then exec R2 else exec R3
    exec R4
until C(R4)>5

```

Пример 6:

```

if A(R1) then
    OAS(R1,R2)
else
    OAS(R3,R4)
while A(RQX1) or not A(DD9) and A(R3)
do
    exec rule_Load
    exec rule_Add
    exec rule_Mult
    exec rule_Store
end

```

Пример 7:

```

while (not A(RQX1) and not A(DD9)) or (C(R3)>5)
repeat
    PRIOR(rule_Mult,rule_Add)
    OCS(rule_Store, rule_Load)
until A(RQX1) or A(DD9)

```

Следует отметить, что язык и система SPARQL/C могут использоваться для создания информационных систем широкого класса в семантическом Web. В данных системах информация хранится в RDF-хранилищах данных, а основные функции по управлению данными возложены на SPARQL-серверы. Основные функции по обработке данных в этом случае можно было бы переложить на SPARQL/C.

3 Инструментальная программная системы для поддержки трансформации графов на основе языка SPARQL

3.1 Выбранные технологии для разработки

В первую очередь необходимо определить язык программирования для разработки ПО. Наиболее подходящим языком программирования является C#. Язык программирования C# — это типизированный, объектно-ориентированный, простой и в то же время мощный язык программирования, который позволяет разработчикам создавать многофункциональные приложения. Инструментарий C# позволяет решать широкий круг задач, язык действительно очень мощный и универсальный. На нём часто разрабатывают: веб-приложения, игры, мобильные приложения для Android или iOS, программы под Windows. Перечень возможностей разработки практически не имеет ограничений благодаря широчайшему набору инструментов и средств.

Для работы с RDF-данными и выполнения SPARQL-запросов был выбран сервер Apache Jena Fuseki. Fuseki предоставляет SPARQL-эндпоинт для выполнения запросов и обновлений данных через HTTP [29]. Поддерживает выполнение всех типов SPARQL-запросов (SELECT, CONSTRUCT, ASK, DESCRIBE) для выборки данных из RDF-хранилища. Позволяет добавлять, обновлять и удалять RDF-данные с помощью SPARQL Update. Является мощным инструментом для работы с семантическими веб-данными и предоставляя богатый набор возможностей для разработки приложений, интеграции данных и проведения семантического анализа.

3.2 Назначение, особенности и функции программы

SPARQL/C-A — это программная система для автоматического выполнения SPARQL UPDATE — запросов. Запуском правил управляет специальная программа (скрипт), написанная на языке SPARQL/C.

Разработанное приложение включает в себя следующие функции:

- ввод информации о SPARQL – сервере (IP-адрес сервера, номер порта) и рабочем наборе данных (имя набора данных);

- управление текстовыми файлами со SPARQL Update-запросами;
- управление RDF-наборам данных:
 - 1) создание нового набора данных;
 - 2) загрузка в него данных из файла;
 - 3) удаление набора данных;
 - 4) просмотр списка наборов данных, существующих на SPARQL-сервере;
 - 5) просмотр выбранного набора данных;
- выполнение SPARQL UPDATE – запросов в пошаговом и автоматическом режиме по заданному алгоритму.

Листинг разработанной программы представлен в приложение А.

3.3 Описание пользовательского интерфейса

Основное меню приложения SPARQL/C-A, изображенное на рисунке 3, обеспечивает пользователю доступ к основным функциям и операциям, необходимым для работы с данными в формате RDF с использованием языка запросов SPARQL. Меню «File» предоставляет функции для работы с текстовыми файлами SPARQL Update-запросов и завершения работы с приложением.

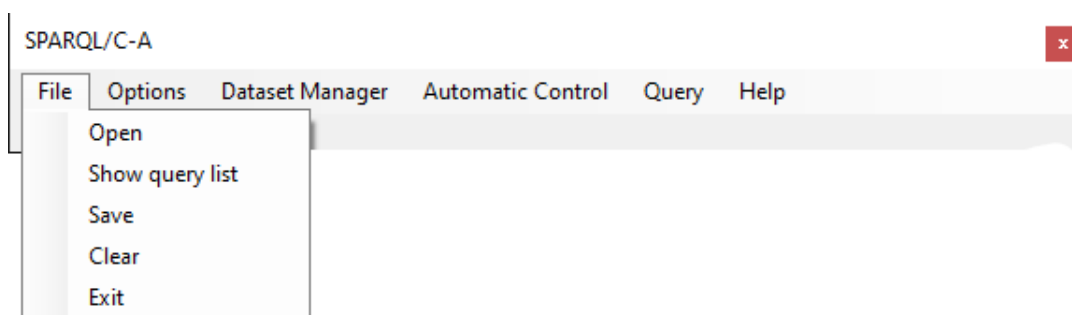


Рисунок 3 – Основное меню приложения

Пункты меню «File»:

- Open – загрузить текстовый файл со SPARQL Update – запросами,
- Show query list – показать список SPARQL Update – запросов,
- Save – сохранить текстовый файл со SPARQL Update – запросами,
- Clear – очистить систему (сброс в исходное состояние),

– Exit – выйти из системы.

Пункт меню (и, соответственно, окно «Options»), представленное на рисунке 4, позволяет пользователю выбрать параметров подключения к SPARQL серверу.

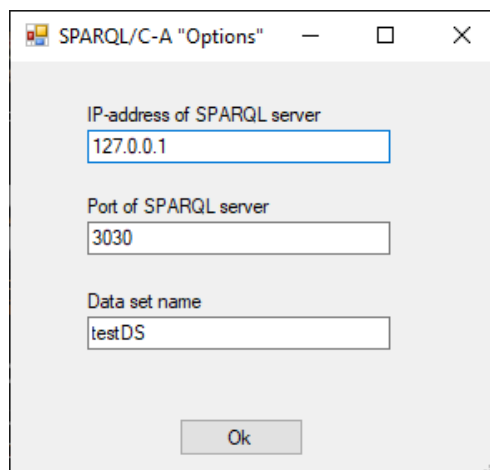
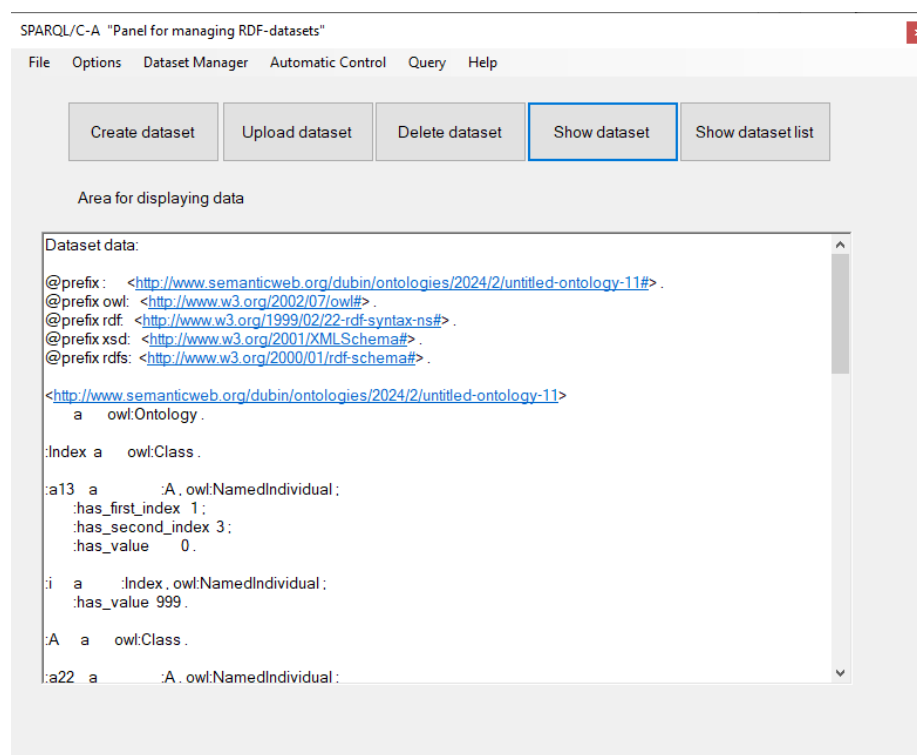


Рисунок 4 – Окно «Options»

Окно «Dataset Manager» предоставляет интерфейс для управления наборами данных на SPARQL-сервере в соответствии с рисунком 5.



```
Dataset data:
@prefix : <http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11#>
  a owl:Ontology .

:Index a owl:Class .

:a13 a owl:NamedIndividual ;
    :has_first_index 1 ;
    :has_second_index 3 ;
    :has_value 0 .

:i a :Index, owl:NamedIndividual ;
    :has_value 999 .

:A a owl:Class .

:a22 a :A, owl:NamedIndividual ;
```

Рисунок 5 – Окно «Dataset Manager»

Окно «Dataset Manager» включает в себя следующие пункты вложенного меню:

- Create dataset – создание нового набора данных на SPARQL-сервере,
- Upload dataset – загрузка в набор данных на SPARQL-сервере данных из файла,
- Delete dataset – удаление набора данных на SPARQL-сервере,
- Show dataset list – просмотр списка наборов данных, существующих на SPARQL-сервере,
- Show dataset – просмотр выбранного набора данных.

Окно «Automatic Control» (рисунок 6) обеспечивает пользователю возможность вводить и выполнять скрипт на языке SPARQL/C. Включает в себя следующие элементы:

- текстовое поле «SPARQL/C script» – в этом поле пользователь может просматривать и редактировать скрипт, при пошаговом выполнении программы следующая в очереди на выполнения команда выделяется цветом (рисунок 7),
- текстовое поле «Messages» - предназначено для вывода хода выполнения скрипта или ошибок считывания, выполнения скрипта,
- кнопка «Load Script» – загрузка скрипта из текстового файла,
- кнопка «Edit Script» – остановка выполнения кода и переход в режим редактирования,
- кнопка «Save script» – сохранение скрипта в текстовый файл,
- кнопка «Execute Step» – нажатие на кнопку выполнит активную в текущий момент операцию. Активная операция выделяется цветом,
- кнопка «Execute All» - нажатие на кнопку выполнит весь скрипт целиком.

Окно «Help» (рисунок 8) позволяет пользователям быстро получить информацию о различных аспектах использования программы. Каждый раздел содержит полезные сведения о том, как использовать функции программы, что они означают и как они работают.

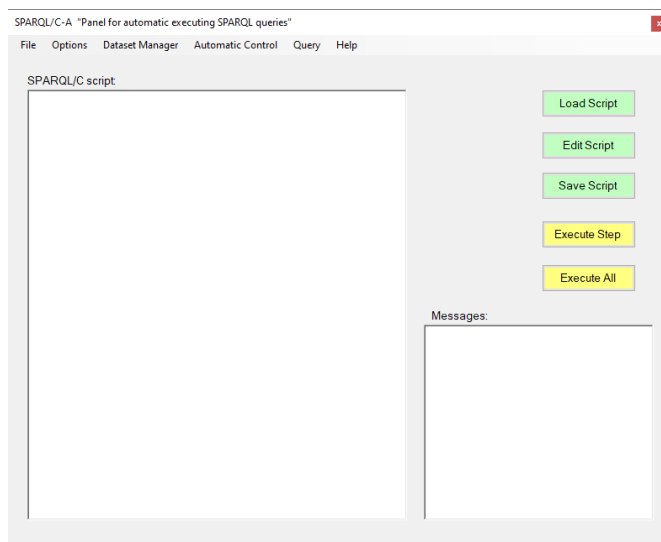


Рисунок 6 – Окно «Automatic control»

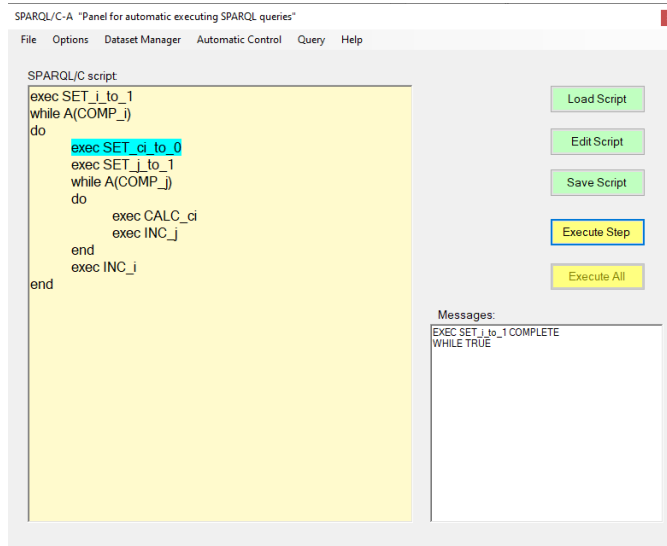


Рисунок 7 – Окно «Automatic control» в пошаговом режиме выполнения

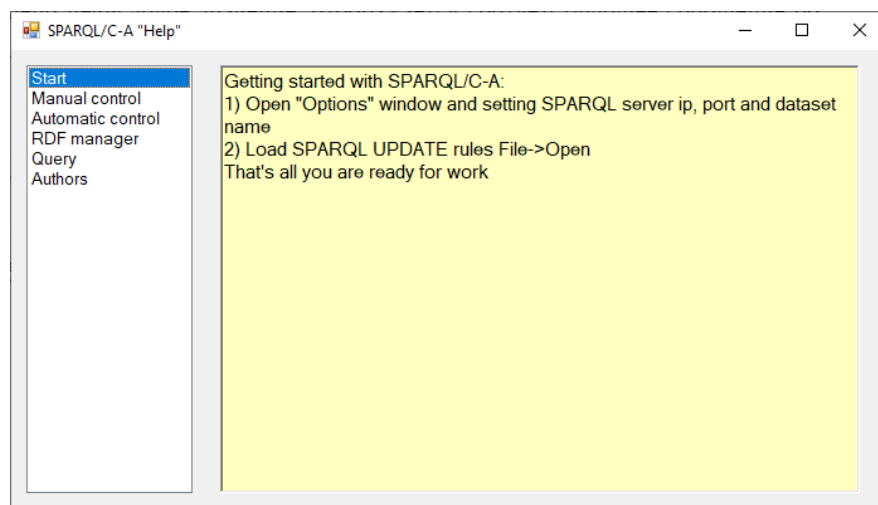


Рисунок 8 – Окно «Help»

3.4 Алгоритм работы интерпретатора

В связи с тем, что любые ошибки в ходе выполнения скрипта могут испортить изначальный набор данных проверка и разбор скрипта выполняется при первом запуске его выполнения.

Разбор скрипта на функциональные составляющие происходит в два этап:

- считывание лексем, выполняется с помощью методов класса `Lexer`,
- формирование из лексем абстрактного синтаксического дерева, реализовано с помощью классов `CodeAnalyzer`, `CodeTree` и `ConditionParser`. На данном этапе все не функциональные операторы удаляются (например, операторы `do-end` удаляются). При встрече условного оператора (`if`, `while`, `until`) условие разбирается на свое дерево.

Для выполнения скрипта из абстрактного синтаксического дерева используется рекурсивный вызов абстрактного метода `Execute` из класса `CodeTree`.

На рисунке 9 представлена диаграмма классов интерпретатора. На рисунках 10, 11 изображены абстрактные классы `CodeTree` и `Node` вместе с их наследниками.

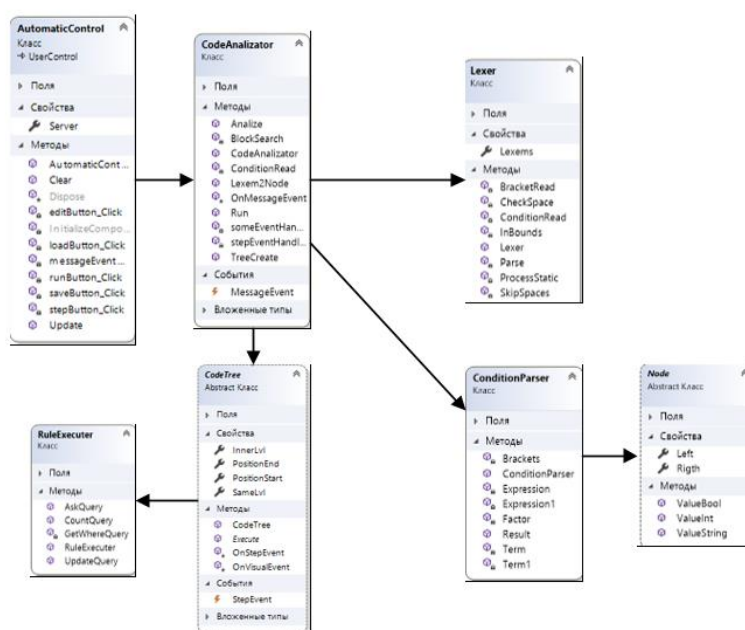


Рисунок 9 – UML – диаграмма классов

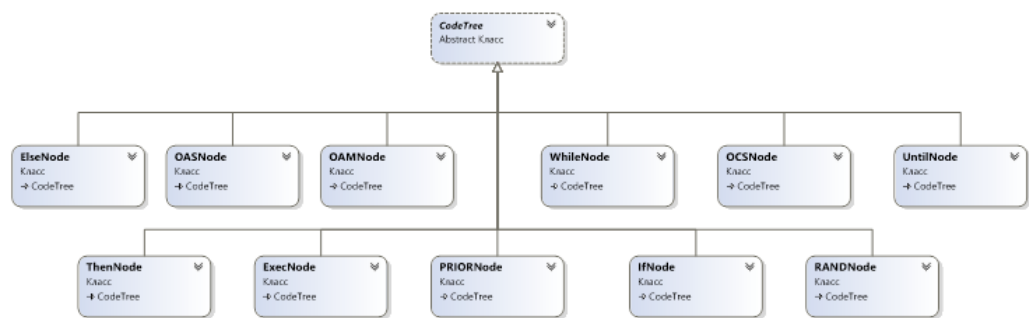


Рисунок 10 – UML – диаграмма классов для наследников класса CodeTree

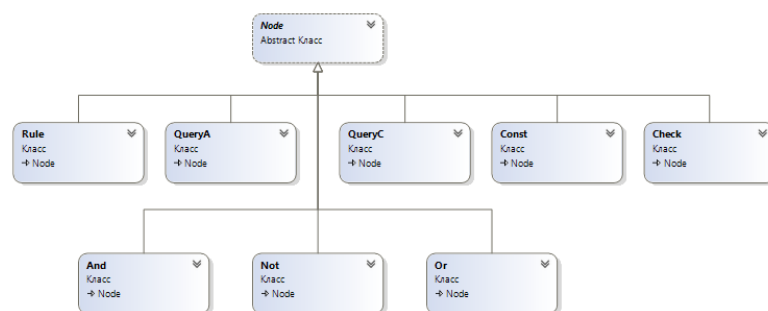


Рисунок 11 – UML – диаграмма классов для наследников класса Node

Класс AutomaticControl предоставляет интерфейс для взаимодействия с пользователем, позволяет загружать, редактировать, запускать программу (скрипт) на языке SPARQL/C. Отвечает за создание экземпляра класса CodeAnalizator.

Класс CodeAnalizator является ключевым компонентом системы, разработанной для разбора и выполнения пользовательского сценария или кода, возможно, связанного с запросами SPARQL. Класс использует лексический анализатор для токенизации входного кода и создает синтаксическое дерево, представляющее структуру программы. Затем он может выполнить код, предоставляя события для различных этапов и сообщений.

Абстрактный класс CodeTree предназначен для формирования абстрактного синтаксического дерева из производных классов. Узлы этого дерева представляют собой структуры управления и действия, такие как while, if, until, exes и модели выполнения (OAS, OAM, OCS, PRIOR, RAND).

Каждый производный класс обладает своей реализацией метода Execute, позволяющий в дальнейшем выполнять программу (скрипт) с помощью рекурсии.

Класс ConditionParser и его вспомогательные классы (Node и производные от него классы) реализуют парсер условий, который позволяет разбирать и оценивать логические выражения. Методы класса используется для анализа строк условий и преобразования их в дерево выражений, которое можно выполнить асинхронно.

Класс RuleExecutor предназначен для выполнения различных типов SPARQL-запросов к серверу. Он включает в себя методы для выполнения запросов обновления (UPDATE), проверки условий (ASK) и подсчета результатов (COUNT). Метод GetWhereQuery используется для создания запросов типа ASK и COUNT. Он позволяет извлекать из запроса часть, начинающуюся на WHERE. Метод UpdateQuery отправляет SPARQL Update-запрос к серверу, и при получении ответа возвращает true или false в зависимости от успеха выполнения запроса. Метод AskQuery отправляет информационный запрос к серверу, и возвращает истину или ложь в зависимости от выполнения условия. Метод CountQuery выполняет запрос к серверу возвращает число найденных подграфов.

4 Примеры реализации конкретных систем трансформации графов с использованием разработанного инструментария

4.1 Решение задачи поиска кратчайшего пути

В данном разделе предлагается реализация графовой грамматики для поиска кратчайших путей в графе Shortest Paths (рисунок 12), представленной в примерах системы AGG, с использованием языка SPARQL 1.1.

Суть задачи сводится к следующему – в графе, представляющем города и дороги, их соединяющие, необходимо найти кратчайший путь из начального города в конечный.

В графе используется два типа узлов: City (города) и State (состояние). Для обозначения обоих типов используется прямоугольник черного цвета. Узлы типа City взвешены атрибутом n (имя узла). Этот атрибут не принимает участие в вычислениях. Узел типа State имеет атрибут max, который используется для хранения максимального (на данный момент) расстояния. Используется пять типов дуг, два типа именованные, остальные различаются цветом и стилем. Дуги типа Start и Goal отмечают начальный и конечный узлы, соответственно. Сплошные черные дуги определяют дороги, они взвешены атрибутом d (расстояние между городами). Пунктирная синяя дуга определяет суммарное расстояние от начального узла до узла-приемника дуги. Точечная красная дуга представляет участок найденного минимального пути, она также взвешена атрибутом d.

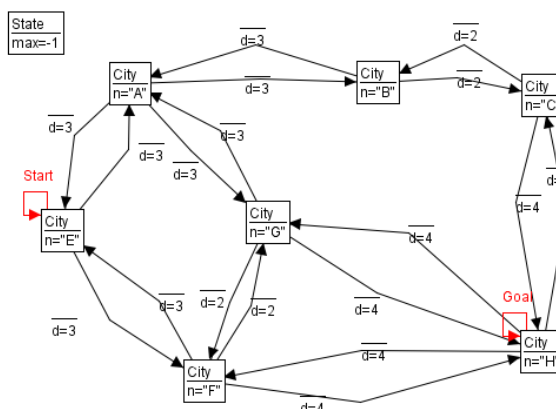


Рисунок 12 – Исходный граф Shortest Path

Соответствующая графовая грамматика включает восемь ПГТ представленных на рисунке 13: [0]start, [1]expandStart, [2]expand, [2]expandMore, [2]setMax, [3]showRoad, [3]showRoad1, [4]removeExpand. Число в квадратных скобках показывает уровень правила. Каждое ПГТ имеет левую и правую части. Кроме того, правило может содержать НАС-и АС-условия. Отрицательное условие применения (НАС-условие) запрещает использование правила. Условие применимости (АС-условие) разрешает использование правила. НАС-условие представляется в виде графа (это своего рода запрещенная фигура), а АС-условие – в виде логического выражения, определенного на множестве атрибутов узлов и дуг графа.

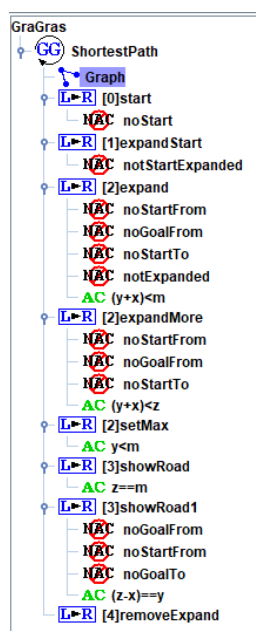


Рисунок 13 – Набор правил трансформации графа Shortest Path

Следует отметить, что AGG-проект ShortestPath.ggx стабильностью не обладает, поскольку при каждом новом запуске в качестве начального и конечного узлов выбираются узлы по случайному принципу. Поэтому данный AGG-проект был модифицирован – а именно в исходном графе была явно указана начальная вершин Е и конечная вершина Н. Первое правило [0]start, назначающее начальную и конечную вершину, было удалено.

В процессе графовых трансформаций начальный граф поэтапно, через ряд промежуточных графов, преобразуется в конечный граф (рисунок 14). В конечном графе будет присутствовать и найденный кратчайший путь (он

показан красными точечными дугами). Статистика по числу выполнения правил приведена ниже: expandStart -2, expand -8, expandMore - 0, setMax - 1, showRoad -1, showRoad1 - 1, removeExpand - 8.

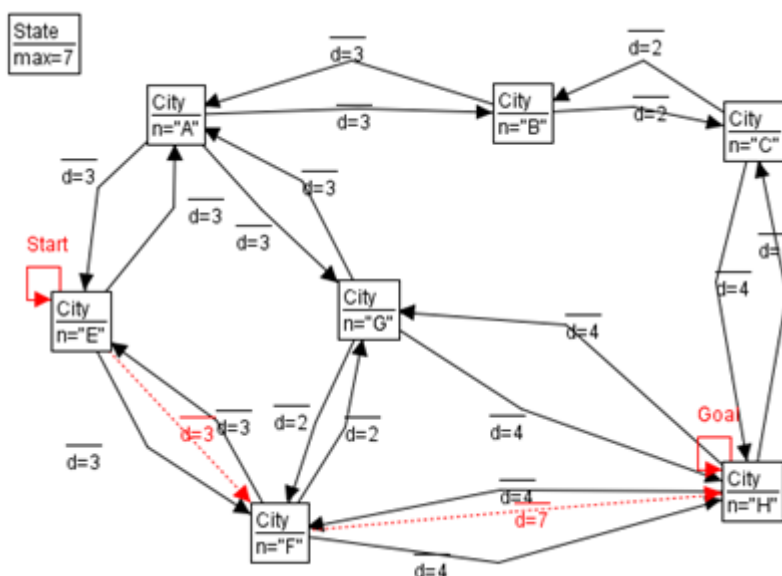


Рисунок 14 - Конечный граф, полученный в результате графовых трансформаций

Для данного примера была сформирована онтология ТАГ в системе Protégé [30]. Характерной особенностью данной онтологии является то, что отдельно не выделяются и не определяются типы вершин, типы дуг и типы атрибутов. Связь вершины или дуги графа с соответствующим типом вершины или дуги определяется через свойство по данным, с помощью которого указывается только имя типа (в виде строковых данных). Для определения атрибута вершины или дуги графа используется только свойство по данным. Данное свойство задает значение атрибута. При этом тип данных атрибута определяется в описании Range (области значений свойства). Имя атрибута задается следующим образом в имени свойства по данным: has_attr_<имя атрибута>. Таким образом, типизация в данной онтологии не используется явно. Все это позволяет значительно упростить онтологию ТАГ и сократить ее объем. Полный текст данной онтология в формате owl представлена в приложение Б.

Классы онтологии:

– Node – узлы (вершины) графа,

– Edge – направленные ребра (дуги) графа.

Объектные свойства:

– comes_from – определяет вершину-источник для дуги,

– comes_to – определяет вершину-приемник для дуги.

Свойства по данным:

– has_attr_n – иметь значение вершинного атрибута n (строкового типа),

– has_attr_d – иметь значение реберного атрибута d (целого типа),

– has_attr_max – иметь значение вершинного атрибута max (целого типа),

– has_node_type – иметь имя типа вершины,

– has_edge_type – иметь имя типа дуги.

Рассмотрим пример реализации на языке SPARQL Update простого трансформационного правила expand. Данное правило определяет минимальные расстояния для узлов, между которыми есть одно ребро и известно предыдущее расстояние. Как видно из рисунка 15 правило включает левую часть (LHS), правую часть (RHS) и четыре NAC-условие (изображено крайним левым графом на рисунке 16).

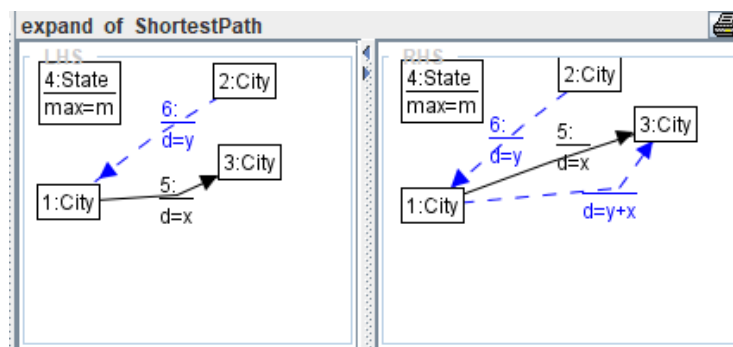


Рисунок 15 – Правило графовой трансформации expand (фрагмент скриншота AGG)

В приложение В представлен SPARQL Update – запрос для данного правила. Левая часть правила образуется WHERE-предложением, так же здесь описываются NAC-условия с помощью четырех FILTER NOT EXISTS конструкций. Правая часть правила описывается INSERT-предложением,

выполняющее добавление новой дуги в случае успешного нахождения подграфа левой части.

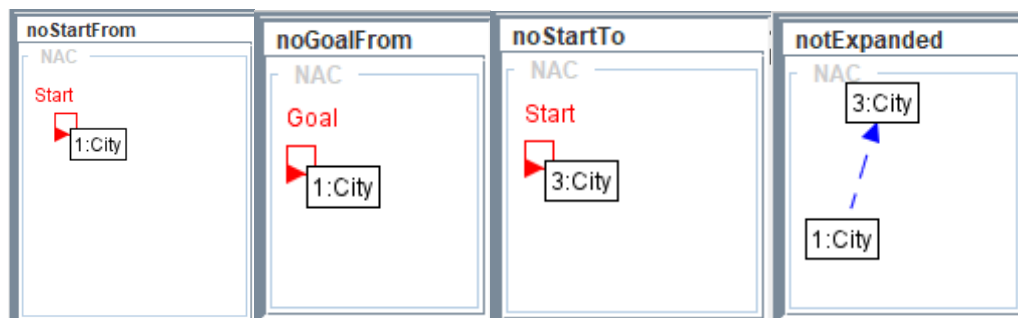


Рисунок 16 – NAC-условия правила графовой трансформации expand (фрагмент скриншота AGG)

SPARQL Update-запросы выполнялись в ручном режиме с использованием Web-интерфейса SPARQL-сервера Apache Jena Fuseki с учетом «поуровневой» модели выполнения правил.

В результате реализации данного примера можно констатировать следующее:

- кратчайший путь, найденный с помощью SPARQL UPDATE-запросов, соответствует кратчайшему пути, найденному системой AGG,
- разработанная система SPARQL UPDATE-запросов правильно реализует систему трансформации графов для поиска кратчайших путей.

Реализация данного примера была опубликована в [31].

4.2 Вычисление суммы целых чисел на языке SPARQL

Для демонстрации использования онтологий и языка SPARQL в арифметических вычислениях, а именно: в вычислении суммы целых чисел, в том числе с использованием языка и описанного инструментального средства рассмотрим следующий пример.

Онтология состоит из двух классов, где Number - это число, а SUM – это сумма в соответствии с рисунком 17.

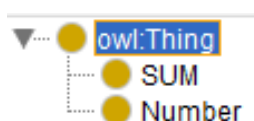


Рисунок 17 – Классы онтологии

Для построения онтологии, описывающей вычислении суммы целых чисел, можно определить следующие свойства по данным (рисунок 18):

- has_value – иметь значение для числа (целое),
- has_been_used – признак использования значения числа при вычислении суммы (false – число не использовалось, true – число было использовано),
- has_total_value – иметь значение для суммы (целое).

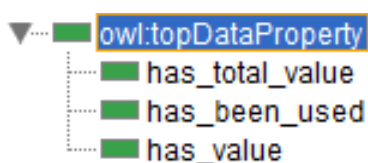


Рисунок 18 – Свойства по данным

В онтологию были добавлены индивиды представленные на рисунке 19, $n1..n5$ соответственно числа со значениями от одного до пяти, а s – это сумма. На рисунке 20 приведен пример описание индивида.

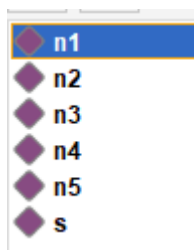


Рисунок 19 – Индивиды

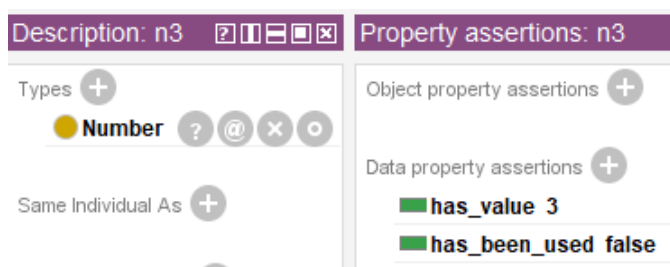


Рисунок 20 – Пример описания индивида $n3$

Текст онтологии для данного примера в формате owl представлен в приложение Г.

Запрос *ADD* – это основной SPARQL Update-запрос в данном примере. Этот запрос добавляет одно конкретное значение числа к сумме. После добавления к сумме число помечается как «использованное».

```

DELETE {
:s :has_total_value ?sum.
?n :has_been_used false}
INSERT {
:s :has_total_value ?newsum.
?n :has_been_used true}
WHERE
{SELECT ?n ?sum ?newsum
WHERE
{?n :has_value ?v.
?n :has_been_used false.
:s :has_total_value ?sum.
BIND (?sum+?v AS ?newsum)}}
LIMIT 1}

```

Запрос *TEST* – это *ASK*-запрос, определяющий есть ли неиспользованные числа для суммирования.

```

ASK
{?n :has_been_used false}

```

Для проверки данного примера проведем ручной тест (рисунок 21).

Запрос для проверки значения суммы представлен ниже.

```

SELECT ?sum
WHERE { :s :has_total_value ?sum}

```

Проверяем начальное значение суммы. Начальное значение суммы равно нулю, что верно. Выполняем запрос *ADD*. Проверяем, какая получилась сумма. Она равна двум. С помощью *ASK*-запроса *TEST* проверяем, есть ли неиспользованные числа для суммирования. Они есть. Значит продолжаем процесс вычисления суммы, повторяя выполнение запроса *ADD* с последующей проверкой неиспользованные числа для суммирования. На пятом шагу вычислений мы получаем в результате пятнадцать, что является верным. Проверка с помощью *ASK*-запроса *TEST* показывает, что неиспользованные числа для суммирования закончились. Полученный результат является итоговым.

Начало	Шаг 1	Шаг 2	Шаг 3	Шаг 4	Шаг 5												
<table><tr><th>sum</th></tr><tr><td>1 "0"<xsd:integer< td=""></xsd:integer<></td></tr></table>	sum	1 "0" <xsd:integer< td=""></xsd:integer<>	<table><tr><th>sum</th></tr><tr><td>1 "2"<xsd:integer< td=""></xsd:integer<></td></tr></table> <div>✓ True</div>	sum	1 "2" <xsd:integer< td=""></xsd:integer<>	<table><tr><th>sum</th></tr><tr><td>1 "7"<xsd:integer< td=""></xsd:integer<></td></tr></table> <div>✓ True</div>	sum	1 "7" <xsd:integer< td=""></xsd:integer<>	<table><tr><th>sum</th></tr><tr><td>1 "10"<xsd:integer< td=""></xsd:integer<></td></tr></table> <div>✓ True</div>	sum	1 "10" <xsd:integer< td=""></xsd:integer<>	<table><tr><th>sum</th></tr><tr><td>1 "14"<xsd:integer< td=""></xsd:integer<></td></tr></table> <div>✓ True</div>	sum	1 "14" <xsd:integer< td=""></xsd:integer<>	<table><tr><th>sum</th></tr><tr><td>1 "15"<xsd:integer< td=""></xsd:integer<></td></tr></table> <div>✗ False</div>	sum	1 "15" <xsd:integer< td=""></xsd:integer<>
sum																	
1 "0" <xsd:integer< td=""></xsd:integer<>																	
sum																	
1 "2" <xsd:integer< td=""></xsd:integer<>																	
sum																	
1 "7" <xsd:integer< td=""></xsd:integer<>																	
sum																	
1 "10" <xsd:integer< td=""></xsd:integer<>																	
sum																	
1 "14" <xsd:integer< td=""></xsd:integer<>																	
sum																	
1 "15" <xsd:integer< td=""></xsd:integer<>																	

Рисунок 21 – Ручное выполнение

Из рассмотренного ранее алгоритма можно получить два варианта программы:

– первый вариант программы на языке графо-трансформационной системы,

```
while A(TEST)
    exec ADD
```

– второй вариант программы на языке графо-трансформационной системы.

```
repeat
    exec ADD
until not A(TEST)
```

Как можно заметить из рисунков 22 и 23 запрос ADD выполнялся пять раз, что соответствует ручному тесту.

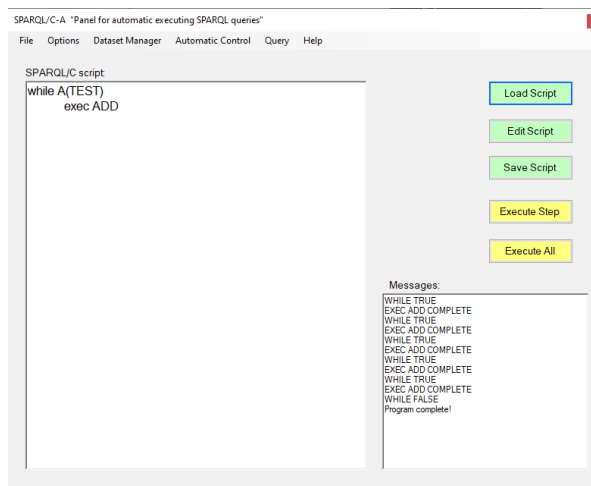


Рисунок 22 – Первый вариант автоматическое вычисление суммы

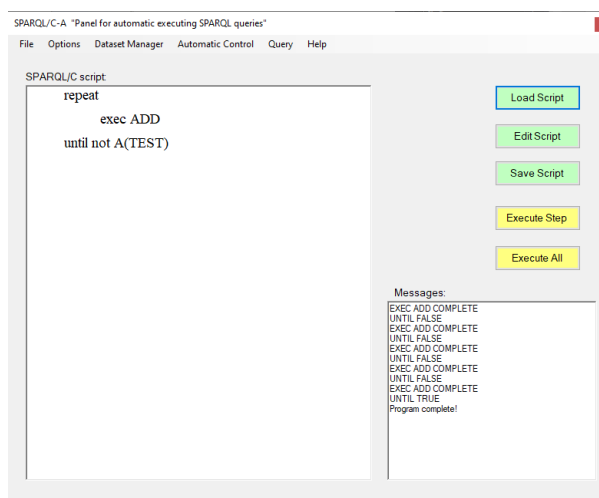


Рисунок 23 – Второй вариант автоматическое вычисление суммы

4.3 Умножение матрицы на вектор на языке SPARQL

Для демонстрации использования онтологий и языка SPARQL в арифметических вычислениях, а именно: умножении матрицы целых чисел на целочисленный вектор, в том числе с использованием языка и описанного инструментального средства рассмотрим следующий пример.

Суть операции умножения матрицы A , состоящей из m строк и n столбцов, на вектор-столбец X , содержащий n компонентов, заключается в следующем:

- матрица A имеет размер $m \times n$. Каждый элемент матрицы обозначается как A_{ij} , где i указывает на номер строки (от 1 до m), а j указывает на номер столбца (от 1 до n),

- вектор X представляет собой столбец с n компонентами, каждая из которых обозначается как X_j , где j меняется от 1 до n ,

- результатом умножения будет вектор Y , состоящий из m компонентов. Каждый элемент этого вектора обозначается как Y_i , где i указывает на номер компонента (от 1 до m).

Каждый компонент результирующего вектора Y находится по формуле:

$$Y_i = \sum_{j=1}^n A_{ij} X_j,$$

где $i = 1, 2, 3, \dots, m$.

Из этой формулы следует внешний цикл по индексу i . В теле этого цикла происходит накопление суммы. Обычно специальной переменной для суммы не используют, а накапливают ее в i -том компоненте вектора Y . Перед накоплением суммы Y_i обнуляется, а затем организуется цикл по индексу суммирования j . Блок-схема этого алгоритма приведена на рисунке 24.

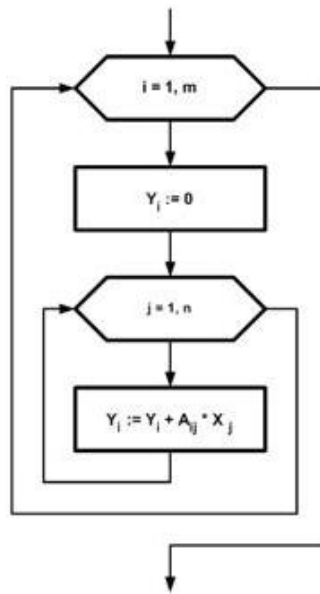


Рисунок 24 – Алгоритм умножения матрицы A на вектор X

Представим алгоритм умножения на более низком уровне, пригодном, чтобы выделить основные операции (команды), которые можно было бы непосредственно реализовать на языке SPARQL (при этом управляющий поток, то есть порядок выполнения операций, реализуется вручную):

```

i=1
M1: c(i)=0
j=1
M2: c(i)= c(i)+a(i,j)*b(j)
j=j+1
ЕСЛИ j≤n, ТО Перейти на метку M2
i=i+1
ЕСЛИ i≤m, ТО Перейти на метку M1
  
```

Выделим из вышеприведенного алгоритма на псевдокоде элементарные действия, которые можно напрямую реализовать на языке SPARQL:

- SET_i_to_1 – установка индекса i в единицу,
- SET_j_to_1 – установка индекса j в единицу,
- SET_ci_to_0 – установка элемента c(i) в ноль,
- INC_i – инкремент индекса i,
- INC_j – инкремент индекса j,
- CALC_ci – вычисление c(i) по формуле $c(i) = c(i) + a(i,j) * b(j)$,

– COMP_i – сравнение индекса i с границей m=2 (на меньше или равно),

– COMP_j – сравнение индекса j с границей n=3 (на меньше или равно).

Онтология для умножения матрицы на вектор содержит четыре класса представленные на рисунке 25:

– A – элементы матрицы A,

– B – элементы вектора-столбца B,

– C – элементы результирующего вектора-столбца C,

– Index – индексы матрицы, используемые в алгоритме.

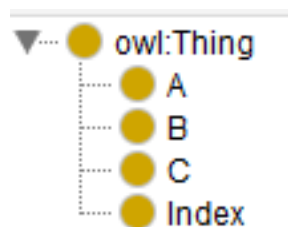


Рисунок 25 – Классы онтологии

Для построения онтологии можно определить следующие свойства по данным (рисунок 26):

– has_first_index, иметь номер первого индекса, например, для элемента матрицы a23 это число 2 (целое),

– has_second_index, иметь номер второго индекса, например, для элемента матрицы a23 это число 3 (целое),

– has_value, иметь значение для элемента матрицы, вектора или индекса (целое).



Рисунок 26 – Свойства по данным

Индивиды онтологии представляют собой элементы a11..a23 матрицы A (размерностью 2 на 3), b1, b2, b3 вектора-столбца (размерностью 3), c1, c2

результатирующего вектора-столбца (размерностью 2) и два индекса i, j . На рисунке 27 изображен пример описания индивида a12.

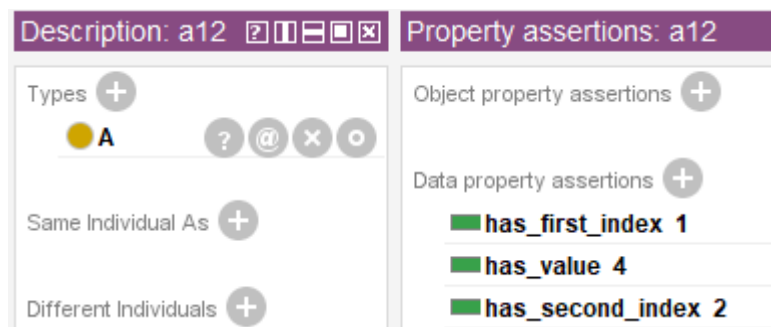


Рисунок 27 – Пример описания индивида a12

Разработанная онтология представлена в приложение Д в формате owl. Реализация элементарных действий в виде SPARQL Update – запросов представлена в приложении Е.

Ниже представлена программа на языке SPARQL/C для умножения матрицы на вектор с помощью цикла repeat-until.

```
exec SET_i_to_1
repeat
  exec SET_ci_to_0
  exec SET_j_to_1
  repeat
    exec CALC_ci
    exec INC_j
  until not A(COMP_j)
  exec INC_i
until not A(COMP_i)
```

Данный алгоритм был протестирован в инструментальной системе SPARQL/C-A (рисунок 28). В результате были получены верные значения нового вектора (рисунок 29).

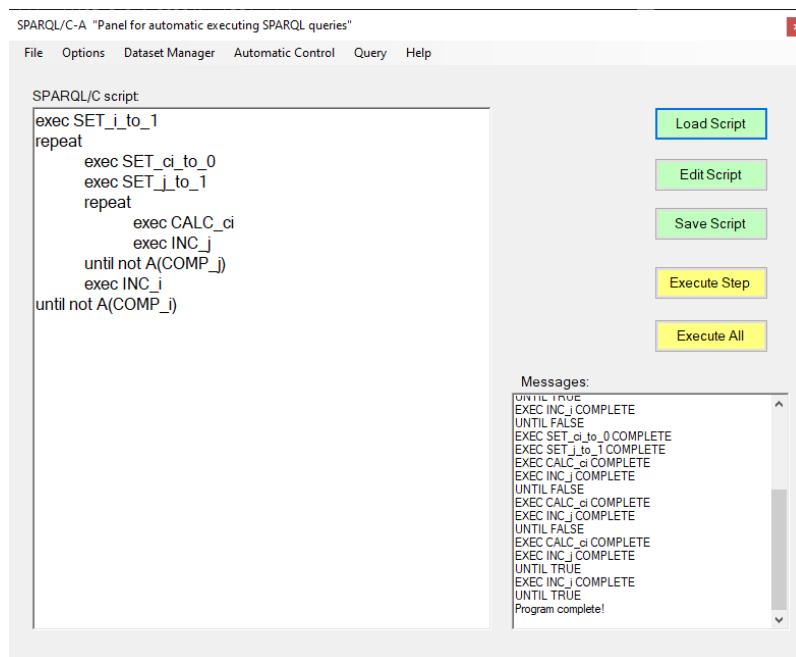


Рисунок 28 – Выполнение алгоритма в системе SPARQL/C-A (цикл repeat-until)

	c	z	vc
1	:c2	"2"^^xsd:integer	"-3"^^xsd:integer
2	:c1	"1"^^xsd:integer	"10"^^xsd:integer

Рисунок 29 – Результат выполнения алгоритма умножения матрицы на вектор

Альтернативное описание алгоритма на языке SPARQL/C с использованием цикла while предложено ниже.

```

exec SET_i_to_1
while A(COMP_i)
do
  exec SET_ci_to_0
  exec SET_j_to_1
  while A(COMP_j)
  do
    exec CALC_ci
    exec INC_j
  end
  exec INC_i
end
  
```

Выполнение данного алгоритма привело к тому же результату что и прошлый алгоритм (рисунок 30).

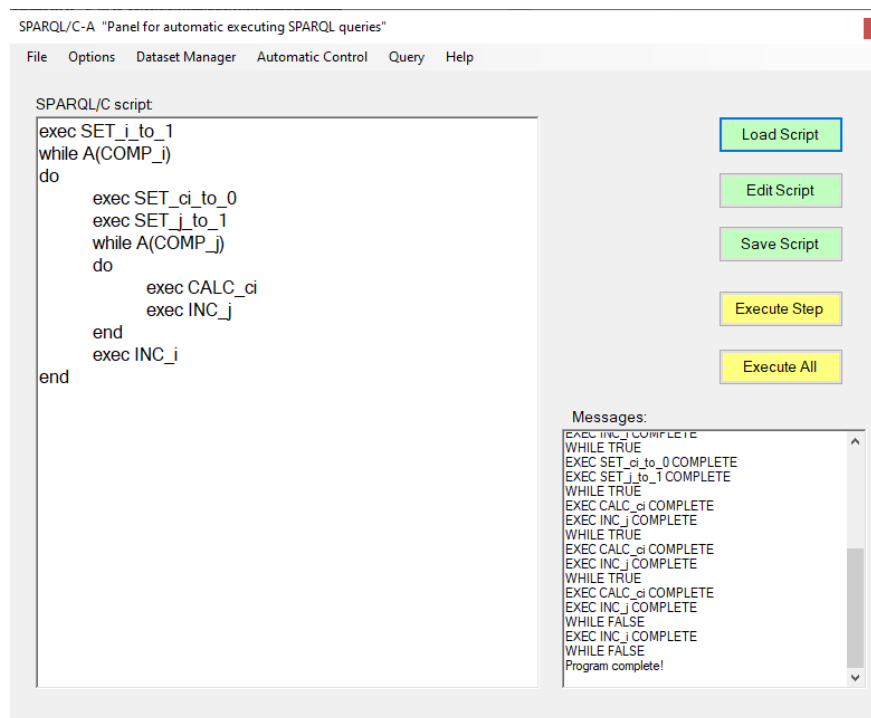


Рисунок 30 – Выполнение алгоритма в системе SPARQL/C-A (цикл while)

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была разработана методика трансформации графов на основе RDF-онтологий и языка SPARQL, позволяющая реализовывать динамические графовые модели.

В результате выполнения данной выпускной квалификационной работы были достигнуты следующие результаты:

- разработана RDF-онтологии типизированных атрибутивных графов (ТАГ), особенностью онтологии является словарь, пригодный для описания любых ТАГ, что позволяет использовать онтологию при разработке автоматизированных систем, включающих обработку графов, на основе технологий семантического Веб,

- разработана методика кодирования правил трансформации графов в виде SPARQL UPDATE запросов. Особенностью методики является возможность одним запросом описывать одновременно и условия выполнения правила преобразования графа, и изменения, применяемые к графу,

- предложены модели выполнения SPARQL Update – правил. Особенностью которых является возможность реализации задач сложной обработки данных, характерной для языков программирования высокого уровня,

- разработан язык описания пользовательских моделей выполнения SPARQL Update – правил, позволяющий разрабатывать произвольные процедуры обработки RDF – данных, что значительно повышает возможности языка SPARQL и позволяет построение информационных обрабатывающих систем в среде семантического Веб,

- разработана инструментальной программной системы для поддержки трансформации графов на основе языка SPARQL. Этот инструмент позволяет проводить трансформацию RDF- онтологий по заданному сценарию. Такой подход позволит пользователям более гибко настраивать выполнение запросов и адаптировать их под конкретные потребности,

- разработаны онтология и правила преобразования для решения задачи поиска кратчайшего пути в графе, позволяющие оценить полезность использования онтологий и языка SPARQL в системах трансформации графов,

- показана применимость разработанной инструментальной программной системы для проведения числовых вычислений с использованием встроенного в эту систему языка описания моделей выполнения SPARQL Update – правил и собственно языка SPARQL.

Дальнейшее развитие методикаи трансформации графов может включать в себя расширение возможностей инструментальной программной системы следующим образом:

- разработка графического редактора SPARQL Update – правил для повышения скорости и эффективности их разработки,

- добавление возможности вызова пользовательских моделей выполнения, наподобие вызова функций в языке С.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Jukss M., Vangheluwe H., Verbrugge C. Implementing Graph Transformation Languages using RDF Storage and SPARQL Queries. – URL: <https://api.semanticscholar.org/CorpusID:9832218> (дата обращения 2022-10-10)
2. Corby O., Faron-Zucker C. A Transformation Language for RDF based on SPARQL // Web Information Systems and Technologies (WEBIST 2015), Lecture Notes in Business Information Processing, vol 246. Springer, Cham, pp 318–340.
3. Duval D., Echahed R., Prost F. An Algebraic Graph Transformation Approach for RDF and SPARQL // Proc. 11th Int. Workshop on Graph Computation Models, GCM 2020. pp. 55-70.
4. Lee S., Sukumar S.S., Hong S., Lim S.-H. Enabling graph mining in RDF triplestores using SPARQL for holistic in-situ graph analysis // Expert Systems with Applications, Vol. 48, 2016, pp. 9-25.
5. Braatz B., Brandt C. Graph Transformations for the Resource Description Framework // Electronic Communications of the EASST, Vol. 10: Graph Transformation and Visual Modeling Techniques 2008. pp. 1-16.
6. Braatz B., Brandt C. How to Modify on the Semantic Web? // Current Trends in Web Engineering. ICWE 2010. Lecture Notes in Computer Science, vol 6385. Springer, 2010. pp 187–198.
7. Suchánek M., Pergl R. Pattern-Based Ontological Transformations for RDF Data using SPARQL // 13th Int. Conf. on Pervasive Patterns and Applications, Porto, Portugal, 2021. pp.11-16.
8. Chabin J., Eichler C., Halfeld-Ferrari M., Hiot N. Graph Rewriting Rules for RDF Database Evolution: Optimizing Side-Eect Processing // International Journal of Web Information Systems. Volume 17, Issue 6. 2021. pp. 1-22.
9. Batarfi O., Elshaw R., Fayoumi A. et al. A distributed query execution engine of big attributed graphs // SpringerPlus 5, Article number: 665, 2016. pp. 1-26.

10. Дубинин В.Н., Королев Б.Д. Реализация графодинамической модели задачи взаимного исключения на языке SPARQL // Информационные технологии в науке и образование. Проблемы и перспективы: сб. ст. по материалам VII всероссийской межвузовской научно-практической конференции. Пенза: Изд-во ПГУ, 2020. – С. 18-20.

11. Дубинин А. В., Ручкин М. А. Учебный пример для изучения технологий RDF-хранилищ данных // Информационные технологии в науке и образовании. Проблемы и перспективы : сборник статей по материалам VIII Всероссийской межвузовской научно-практической конференции, Пенза, 17 марта 2021 года. – Пенза: Пензенский государственный университет, 2021. – С. 135-137.

12. Дубинин А.В., Ручкин М.А., Дубинин В.Н. Моделирование производственных систем с использованием языка SPARQL // Сборник научных статей Международной научно-технической конференции "Современные информационные технологии", Пенза, 2023, вып. 38. – С. 41-55.

13. Ehrig, H. Fundamental theory for typed attributed graph transformation / H. Ehrig, U. Prange, G. Taenzer // Graph Transformation: 2nd Int. Conf. (ICGT 2004). Lecture Notes in Computer Science. – Berlin; Heidelberg : Springer Verlag, 2004. – Vol. 3256. – P. 161–177.

14. Handbook of Graph Grammars and Computing by Graph Transformations / G. Rozenberg (ed.) // World Scientific. – 1997. – Vol. 1.

15. Taentzer G. AGG: A graph transformation environment for modeling and validation of software // Applications of Graph Transformations with Industrial Relevance (AGTIVE), LNCS, vol 3062., 2004, pp. 446-453.

16. AGG Homepage. – URL: <https://www.user.tu-berlin.de/o.runge/agg/> (дата обращения 2023-02-02)

17. GROOVE Homepage.– URL: <https://groove.ewi.utwente.nl> (дата обращения 2023-02-02)

18. Agrawal A. Graph rewriting and transformation (GReAT): A solution for the model integrated computing (mic) bottleneck // Int. Conf. On Automated Software Engineering, 2003.
19. GrGen Homepage. – URL: <http://www.info.uni-karlsruhe.de/software/grgen/> (дата обращения 2023-02-02)
20. AToM3 Homepage. – URL: <http://atom3.cs.mcgill.ca> (дата обращения 2023-02-02)
21. The Object Management Group. – URL: <http://www.omg.org/spec/QVT/1.1/> (дата обращения 2023-02-02)
22. Jouault F., Allilaire F., Bezivin J., Kurtev I., and Valduriez P. ATL: a qvt-like transformation language // OOPSLA Companion, 2006, pp. 719-720.
23. Csertan G., Huszerl G., Majzik I., Pap Z., Pataricza A., Varro D. Viatra “Visual automated transformations for formal verification and validation of uml models” // Proc. of the 17th IEEE int. conf. On Automated software engineering, Washington, USA, 2002, pp 267.
24. The FUJABA Toolsuite. – URL: <https://web.cs.upb.de/archive/fujaba/> (дата обращения 2023-02-02)
25. Schurr A., Winter A.J., Zundorf A. The PROGRES approach: language and environment // Handbook of graph grammars and computing by graph transformation: applications, languages, and tools. vol 2. World Scientific Publishing Co., Inc. 1999. pp. 487– 550.
26. SPARQL Query Language for RDF. – URL: <https://www.w3.org/TR/rdf-sparql-query/> (дата обращения 2024-04-04)
27. Ручкин А.М., Дубинин А.В., Дубинин В.Н. Онтология типизированных атрибутных графов // Сб. ст. XI Ежегодн. всеросс. межвуз. науч.-практ. конф. «Информационные технологии в науке и образовании. Проблемы и перспективы (ВМНПК – 2024)», Пенза, 2024.
28. Ehrig H. Attributed graphs and typing: Relationship between different representations // Bull. Eur. Assoc. Theor. Comput. Sci. (EATCS), N 82, 2004. - P. 175-190.

29. Apache Jena Fuseki. – URL: <https://jena.apache.org/documentation/fuseki2/> (дата обращения 2023-04-04)

30. Protégé Homepage. – URL: <https://protege.stanford.edu> (дата обращения 2024-05-04)

31. Дубинин В.Н., Ручкин М.А., Дубинин А.В., Климкина Л.П. Использование языка SPARQL в реализации графодинамической модели для поиска кратчайших путей в графе // Современные технологии в науке и образовании – СТНО-2024 [текст]: сб. тр. VII междунар. науч.-техн. форума: в 10 т. Т.5./ под общ. ред. О.В. Миловзорова. – Рязань: Рязан. гос. радиотехн. ун-т, 2024. – С. 160-165.

ПРИЛОЖЕНИЕ А

(Обязательное)

Реализация распределенной графо-трансформационной
системы на основе онтологий и языка SPARQL

Листинг программы

Файл MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace SparqlClient
{
    public struct ServerDataType
    {
        public string IPAddress;
        public string Port;
        public string DatasetName;
    }

    public partial class MainForm : Form
    {
        public List<QueryType> queryList;
        public string prefixList;
        public ServerDataType ServerData;
        private HelpForm dialog;

        public MainForm()
        {
            InitializeComponent();
            queryList = new List<QueryType>();
            prefixList = string.Empty;
            ServerData.IPAddress = "127.0.0.1";
            ServerData.Port = "3030";
            ServerData.DatasetName = string.Empty;
            this.seriesControl.Server = ServerData;
            this.queryControl.Server = ServerData;
            dialog = new HelpForm();
        }

        public ServerDataType Server
        {
            get { return ServerData; }
            set { ServerData = value; }
        }

        //Обработка нажатия кнопок панели инструментов главного
        окна
        private void optionToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            var option = new OptionForm(this);
```

```

        option.ShowDialog();
        this.seriesControl.Server = ServerData;
        this.queryControl.Server = ServerData;
        this.automaticControl.Server = ServerData;
    }

    private void openToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        var fileContent = string.Empty;
        var filePath = string.Empty;

        using (OpenFileDialog openFileDialog = new
OpenFileDialog())
        {
            //openFileDialog.InitialDirectory = "c:\\";
            openFileDialog.Filter = "txt files
(*.txt)|*.txt|All files (*.*)|*.*";
            openFileDialog.FilterIndex = 2;
            openFileDialog.RestoreDirectory = true;

            if (openFileDialog.ShowDialog() ==
DialogResult.OK)
            {
                //Get the path of specified file
                filePath = openFileDialog.FileName;

                //Read the contents of the file into a
stream
                var fileStream = openFileDialog.OpenFile();

                using (StreamReader reader = new
StreamReader(fileStream))
                {
                    fileContent = reader.ReadToEnd();
                    string[] fileContentLine =
fileContent.Split(new string[] { "\r\n", "\n", "\r" },
StringSplitOptions.RemoveEmptyEntries);
                    //DataList.Clear();
                    bool readPrefix = false;
                    bool readQuery = false;
                    QueryType element = new QueryType();
                    foreach (string line in fileContentLine)
                    {
                        //начало считывание префиксов
                        if (line.StartsWith("#PREFIX"))
                        {
                            readQuery = false;
                            readPrefix = true;
                            continue;
                        }
                        //начало считывание запроса
                        if (line.StartsWith("#QUERY"))

```

```

        {
            if(readQuery)
            {
                queryList.Add(element);
            }
            readPrefix = false;
            readQuery = true;
            string[] stringValues =
line.Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
            element = new QueryType();
            element.Name = stringValues[1];
            foreach (QueryType query in
queryList)
                if (query.Name ==
element.Name)
                {
                    DialogResult result =
MessageBox.Show(
                        "Query " +
query.Name + " already exist. Rewrite?",
                        "Warning",
MessageBoxButtons.YesNo,
MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1,
MessageBoxOptions.DefaultDesktopOnly);
                    if (result ==
DialogResult.Yes)
                    {
                        queryList.Remove(query);
                    }
                    else
                        readQuery = false;
                    break;
                }
            for (int i = 2; i <
stringValues.Count(); i++)
            {
                element.Comment +=
stringValues[i] + " ";
            }
            continue;
        }
        //обнаружен конец файла с правилами
        if (line.StartsWith("#END"))
        {
            if (readQuery)
            {

```

```

        queryList.Add(element);
    }

this.seriesControl.Update(prefixList, queryList);

this.automaticControl.Update(prefixList, queryList);
        break;
    }
    if (readPrefix)
    {
        prefixList +=
SearchPrefix(line);
    }
    if (readQuery)
    {
        element.Data += line + "\r\n";
    }
    }
}

}

}

private string SearchPrefix(string nPrefix)
{
    string result = string.Empty;
    string[] lines = nPrefix.Split(new char[] { '\r',
'\n' }, StringSplitOptions.RemoveEmptyEntries);
    foreach (string str in lines)
        if (!prefixList.Contains(str))
            result += str + "\r\n";
    return result;
}

private void clearToolStripMenuItem_Click(object sender,
EventArgs e)
{
    queryList.Clear();
    prefixList = string.Empty;
    this.seriesControl.Clear();
    this.queryControl.Clear();
    this.automaticControl.Clear();
    ServerData.IpAddress = "127.0.0.1";
    ServerData.Port = "3030";
    ServerData.DatasetName = string.Empty;
    this.seriesControl.Server = ServerData;
    this.queryControl.Server = ServerData;
    this.automaticControl.Server = ServerData;
    this.Text = "SPARQL/C-A";
    this.Controls.Remove(this.rdfManager);
    this.Controls.Remove(this.queryControl);

```

```

        this.Controls.Remove(this.seriesControl);
        this.Controls.Remove(this.automaticControl);
        this.Size = new System.Drawing.Size(590, 83);
    }

    private void exitToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        this.Close();
    }

    private void queryToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        this.Controls.Remove(this.seriesControl);
        this.Controls.Remove(this.rdfManager);
        this.Controls.Remove(this.automaticControl);
        this.Controls.Add(this.queryControl);
        this.Text = "SPARQL/C-A  \"Panel for executing
SPARQL queries\"";
        this.Size = new System.Drawing.Size(800, 650);
        this.queryControl.Server = ServerData;
    }

    private void
datasetManagerToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        this.Controls.Remove(this.seriesControl);
        this.Controls.Remove(this.queryControl);
        this.Controls.Remove(this.automaticControl);
        this.Controls.Add(this.rdfManager);
        this.Text = "SPARQL/C-A  \"Panel for managing RDF-
datasets\"";
        this.Size = new System.Drawing.Size(800, 650);
    }

    private void showQueryListToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        QueryEditorForm dialog = new
QueryEditorForm(prefixList, queryList);
        dialog.ShowDialog();
        prefixList = dialog.PrefixList;
        queryList = dialog.QueryData;
        this.automaticControl.Update(prefixList, queryList);
        this.seriesControl.Update(prefixList, queryList);
    }

    private void
automaticControlToolStripMenuItem_Click(object sender, EventArgs
e)
    {

```

```

        this.automaticControl.Update(prefixList, queryList);
        this.Controls.Remove(this.seriesControl);
        this.Controls.Remove(this.queryControl);
        this.Controls.Remove(this.rdfManager);
        this.Controls.Add(this.automaticControl);
        this.Text = "SPARQL/C-A  \"Panel for automatic
executing SPARQL queries\"";
        this.Size = new System.Drawing.Size(800, 650);
        this.automaticControl.Server = ServerData;
    }

    private void saveToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        using (SaveFileDialog saveFileDialog = new
SaveFileDialog())
        {
            saveFileDialog.Filter = "Text files(*.txt) |
*.txt |All files(*.*)|*.*";
            if (saveFileDialog.ShowDialog() ==
DialogResult.Cancel)
                return;
            string filename = saveFileDialog.FileName;
            string saveFile = "#PREFIX\n" + prefixList;
            foreach (QueryType query in queryList)
                saveFile += "#QUERY " + query.Name + " " +
query.Comment + "\n" + query.Data + "\n";
            saveFile += "#END\n";
            System.IO.File.WriteAllText(filename, saveFile);
        }
    }

    private void helpToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        if (dialog.IsAccessible)
            return;
        if (dialog.IsDisposed)
        {
            dialog = new HelpForm();
        }
        dialog.Show();
    }
}
}

```

Файл OptionForm.cs

```

using System;
using System.Windows.Forms;
using System.Net;
namespace SparqlClient
{

```

```

public partial class OptionForm : Form
{
    MainForm link;
    public OptionForm(MainForm parent)
    {
        InitializeComponent();
        link = parent;
        ErrorLabel.Text = string.Empty;
        IpError.Text = string.Empty;
        PortError.Text = string.Empty;
        DatasetError.Text = string.Empty;

        ServerDataType ServerData = link.Server;
        IpBox.Text = ServerData.IpAddress;
        PortBox.Text = ServerData.Port;
        DatasetBox.Text = ServerData.DatasetName;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        ErrorLabel.Text = string.Empty;
        IpError.Text = string.Empty;
        PortError.Text = string.Empty;
        DatasetError.Text = string.Empty;

        if (string.IsNullOrEmpty(IpBox.Text))
        {
            IpError.Text = "*";
            ErrorLabel.Text = "Empty IP address";
        }
        else if (string.IsNullOrEmpty(PortBox.Text))
        {
            PortError.Text = "*";
            ErrorLabel.Text = "Empty Port";
        }
        else if (string.IsNullOrEmpty(DatasetBox.Text))
        {
            DatasetError.Text = "*";
            ErrorLabel.Text = "Empty Port";
        }
        else if (IpBox.Text.Split(new char[] { '.' },
StringSplitOptions.RemoveEmptyEntries).Length == 4)
        {
            IPAddress ipAddr;
            if (IPAddress.TryParse(IpBox.Text, out ipAddr))
            {
                int n;
                // IP is valid
                if (int.TryParse(PortBox.Text, out n))
                {
                    ServerDataType ServerData;
                    ServerData.IpAddress = IpBox.Text;
                    ServerData.Port = PortBox.Text;
                }
            }
        }
    }
}

```

```

        ServerData.DatasetName =
DatasetBox.Text;

        link.Server = ServerData;
        this.Close();
    }
    else
    {
        PortError.Text = "*";
        ErrorLabel.Text = "Bad Port";
    }
}
else
{
    // invalid IP
    IpError.Text = "*";
    ErrorLabel.Text = "Bad IP address";
}

}
else
{
    // invalid IP
    IpError.Text = "*";
    ErrorLabel.Text = "Bad IP address";
}
}
}
}
}

```

Файл AutomaticControl.cs

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace SparqlClient
{
    public partial class AutomaticControl : UserControl
    {

        string prefixStr;
        List<QueryType> queryData;
        ServerDataType ServerData;

        bool step_run = false;
        public AutomaticControl()
        {
            InitializeComponent();
        }
        public ServerDataType Server
    }
}

```

```

        {
            set
            {
                ServerData = value;
            }
        }
    private void stepButton_Click(object sender, EventArgs
e)
    {
        if (!step_run)
        {
            try
            {
                runButton.Enabled = false;
                logBox.Text = string.Empty;
                scriptBox.BackColor = Color.LemonChiffon;
                scriptBox.ReadOnly = true;
                CodeAnalyzer analyzer = new
CodeAnalyzer(prefixStr, queryData, ServerData);
                analyzer.MessageEvent +=
messageEventHandler;
                analyzer.Analyze(scriptBox.Text);
                CodeAnalyzer.step = 0;
                CodeAnalyzer.run = false;
                CodeAnalyzer.exit = false;
                step_run = true;
                analyzer.Run();
            }
            catch
            {
                logBox.Text += "Run error\n";
                scriptBox.ReadOnly = false;
                runButton.Enabled = true;
                stepButton.Enabled = true;
                scriptBox.BackColor = Color.White;
            }
        }
        else
        {
            if (CodeAnalyzer.step == 0)
            {
                CodeAnalyzer.step++;
            }
        }
    }

    private void runButton_Click(object sender, EventArgs e)
    {
        scriptBox.BackColor = Color.LemonChiffon;
        logBox.Text = string.Empty;

```

```

        try
        {
            CodeAnalyzer analyzer = new
CodeAnalyzer(prefixStr, queryData, ServerData);
            analyzer.MessageEvent += messageEventHandler;
            analyzer.Analyze(scriptBox.Text);
            runButton.Enabled = false;
            stepButton.Enabled = false;
            CodeAnalyzer.step = 0;
            CodeAnalyzer.run = true;
            CodeAnalyzer.exit = false;
            analyzer.Run();
        }
        catch
        {
            logBox.Text += "Run error\n";
            scriptBox.ReadOnly = false;
            runButton.Enabled = true;
            stepButton.Enabled = true;
            scriptBox.BackColor = Color.White;
        }
    }
}

```

e)

```

    {
        CodeAnalyzer.exit = true;
        stepButton.Enabled = true;
        scriptBox.ReadOnly = false;
        runButton.Enabled = true;
        scriptBox.BackColor = Color.White;
        scriptBox.Select(0, scriptBox.TextLength);
        scriptBox.SelectionBackColor = Color.White;
        step_run = false;
    }

    public void Update(string prefixList, List<QueryType>
queryData)
    {
        this.prefixStr = prefixList;
        this.queryData = queryData;
    }

    public void Clear()
    {
        CodeAnalyzer.exit = true;
        stepButton.Enabled = true;
        scriptBox.ReadOnly = false;
        runButton.Enabled = true;
        scriptBox.BackColor = Color.White;
        step_run = false;
    }
}

```

```

        scriptBox.Text = string.Empty;
        logBox.Text = string.Empty;
    }

    private void loadButton_Click(object sender, EventArgs
e)
    {
        var fileContent = string.Empty;
        var filePath = string.Empty;

        using (OpenFileDialog openFileDialog = new
OpenFileDialog())
        {
            //openFileDialog.InitialDirectory = "c:\\";
            openFileDialog.Filter = "txt files
(*.txt)|*.txt|All files (*.*)|*.*";
            openFileDialog.FilterIndex = 2;
            openFileDialog.RestoreDirectory = true;

            if (openFileDialog.ShowDialog() ==
DialogResult.OK)
            {
                //Get the path of specified file
                filePath = openFileDialog.FileName;

                //Read the contents of the file into a
stream
                var fileStream = openFileDialog.OpenFile();

                using (StreamReader reader = new
StreamReader(fileStream))
                {
                    fileContent = reader.ReadToEnd();
                    scriptBox.Text = fileContent;
                }
            }
        }

        private void saveButton_Click(object sender, EventArgs
e)
        {
            using (SaveFileDialog saveFileDialog = new
SaveFileDialog())
            {
                saveFileDialog.Filter = "Text files (*.txt) |
*.txt |All files (*.*)|*.*";
                if (saveFileDialog.ShowDialog() ==
DialogResult.Cancel)
                    return;
                string filename = saveFileDialog.FileName;
                string saveFile = scriptBox.Text;

```

```

        System.IO.File.WriteAllText(filename, saveFile);
    }
}

private void messageEventHandler(object sender,
CustomEventArgs e)
{
    if (e.Message == "eof")
    {
        stepButton.Enabled = true;
        scriptBox.ReadOnly = false;
        runButton.Enabled = true;
        scriptBox.BackColor = Color.White;
        scriptBox.Select(0, scriptBox.TextLength);
        scriptBox.SelectionBackColor = Color.White;
        step_run = false;
        logBox.Text += "Program complete!" + "\n";
    }
    else if (e.Message == "visual")
    {
        scriptBox.Select(0, scriptBox.TextLength);
        scriptBox.SelectionBackColor =
Color.LemonChiffon;
        scriptBox.Select(e.PositionStart, e.PositionEnd
- e.PositionStart);
        scriptBox.SelectionBackColor = Color.Aqua;
        scriptBox.Select(0, 0);
    }
    else
        logBox.Text += e.Message + "\n";
    }
}
}

```

Файл CodeAnalizator.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace SparqlClient
{
    class CodeAnalizator
    {
        public static int step;
        public static bool run;
        public static bool exit;

        public delegate void MessageEventHandler(object sender,
CustomEventArgs e);
        public event MessageEventHandler MessageEvent;
    }
}

```

```

        // public delegate void StepEventHandler(object sender,
CustomEventArgs e);
        // public event StepEventHandler StepEvent;

        string prefixList;
        List<QueryType> queryData;
        ServerDataType ServerData;

        //code data
        Lexer lx;
        CodeTree root;
        public CodeAnalizator(string prefixStr, List<QueryType>
queryData, ServerDataType ServerData)
        {
            root = null;
            this.prefixList = prefixStr;
            this.queryData = queryData;
            this.ServerData = ServerData;

        }

        private void someEventHandler(object sender,
CustomEventArgs e)
        {

        }

        public void Analize(string code)
        {
            string codeStr = code;
            try
            {
                lx = new Lexer(codeStr);
                TreeCreate(lx);
                if (root == null)
                    throw new Exception(string.Format("Code
analyze error: root null"));
            }
            catch (Exception e)
            {
                OnMessageEvent(new CustomEventArgs(e.Message));
                throw;
            }
            return;
        }

        public async void Run()
        {
            try
            {
                await root.Execute();
            }
        }
    }

```

```

        OnMessageEvent(new
CustomEventArgs (string.Format("eof")));
    }
    catch (Exception e) {
        OnMessageEvent(new
CustomEventArgs (string.Format(e.Message)));
    }

}

public void TreeCreate(Lexer lx)
{
    int offs;
    List<Lexem> lxList = lx.Lexems.ToList();
    (root, offs) = Lexem2Node(lxList, 0,
lxList.Count);
}

public (CodeTree, int) Lexem2Node(List<Lexem> lxList,
int start, int end, bool oneRead = false)
{
    CodeTree rt = null, currentNode = null, newNode =
null;

    int i = start;
    int offs;
    for (i = start; i < end; i++)
    {
        if (oneRead && i != start)
            return (rt , (i - start));

        if (lxList[i].Kind == LexemKind.While)
        {
            newNode = new WhileNode(lxList[i].Value,
lxList[i].Offset, lxList[i].Length, prefixList, queryData,
ServerData);

            newNode.StepEvent += stepEventHandler;
            if (lxList[i + 1].Kind == LexemKind.Repeat)
            {
                int inner_end = BlockSearch(lxList, i +
1, end, LexemKind.Repeat, LexemKind.Until);
                (newNode.InnerLvl, offs) =
Lexem2Node(lxList, i + 1, inner_end);
                i = inner_end;
            }
            else if (lxList[i + 1].Kind == LexemKind.Do)
            {
                int inner_end = BlockSearch(lxList, i +
1, end, LexemKind.Do, LexemKind.End);
                (newNode.InnerLvl, offs) =
Lexem2Node(lxList, i + 2, inner_end);
                i = inner_end;
            }
            else

```

```

        {
            (newNode.InnerLvl, offs) =
Lexem2Node(lxList, i + 1, end, true);
            i+= offs;
        }

        //если if - repeat - do то вызывать
Lexem2Node иначе добавит
        if (rt == null) rt = currentNode = newNode;
        else
        {
            currentNode.SameLvl = newNode;
            currentNode = newNode;
        }
        continue;
    }
    else if (lxList[i].Kind == LexemKind.If)
    {
        if((i+1)<end)
            if (lxList[i + 1].Kind ==
LexemKind.Then)
                {
                    newNode = new
IfNode(lxList[i].Value, lxList[i].Offset, lxList[i].Length,
prefixList, queryData, ServerData);
                    newNode.StepEvent +=
stepEventHandler;
                    //newNode.InnerLvl =
Lexem2Node(lxList, i + 1, end, true);
                    if (rt == null) rt = currentNode =
newNode;
                    else
                    {
                        currentNode.SameLvl = newNode;
                        currentNode = newNode;
                    }
                    i++;
                    // then
                    newNode = new
ThenNode(lxList[i].Offset, lxList[i].Length);
                    newNode.StepEvent +=
stepEventHandler;
                    if (lxList[i + 1].Kind ==
LexemKind.Repeat)
                        {
                            int inner_end =
BlockSearch(lxList, i + 1, end, LexemKind.Repeat,
LexemKind.Until);
                            (newNode.InnerLvl, offs) =
Lexem2Node(lxList, i + 1, inner_end);
                            i = inner_end;
                        }
                    }
                }
    }

```

```

else if (lxList[i + 1].Kind ==
LexemKind.Do)
{
    int inner_end =
BlockSearch(lxList, i + 1, end, LexemKind.Do, LexemKind.End);
    (newNode.InnerLvl, offs) =
Lexem2Node(lxList, i + 2, inner_end);
    i = inner_end;
}
else
{
    (newNode.InnerLvl, offs) =
Lexem2Node(lxList, i + 1, end, true);
    i+=offs;
}
currentNode.InnerLvl = newNode;
//else
if (i + 1 < end)
    if (lxList[i + 1].Kind ==
LexemKind.Else)
    {
        i++;
        newNode = new
ElseNode(lxList[i].Offset, lxList[i].Length);
        newNode.StepEvent +=
stepEventHandler;
        if (lxList[i + 1].Kind ==
LexemKind.Repeat)
        {
            int inner_end =
BlockSearch(lxList, i + 1, end, LexemKind.Repeat,
LexemKind.Until);
            (newNode.InnerLvl, offs)
= Lexem2Node(lxList, i + 1, inner_end);
            i = inner_end;
        }
        else if (lxList[i + 1].Kind
== LexemKind.Do)
        {
            int inner_end =
BlockSearch(lxList, i + 1, end, LexemKind.Do, LexemKind.End);
            (newNode.InnerLvl, offs)
= Lexem2Node(lxList, i + 2, inner_end);
            i = inner_end;
        }
        else
        {
            (newNode.InnerLvl, offs)
= Lexem2Node(lxList, i + 1, end, true);
            i+=offs;
        }
        currentNode.InnerLvl.SameLvl
= newNode;

```

```

        }

        continue;
    }
    else
        throw new
Exception(string.Format("Error in if-then construction"));
    else
        throw new Exception(string.Format("Error
in if-then construction"));
    }
    else if (lxList[i].Kind == LexemKind.Repeat)
    {
        int inner_end = BlockSearch(lxList, i, end,
LexemKind.Repeat, LexemKind.Until);
        newNode = new
UntilNode(lxList[inner_end].Value, lxList[inner_end].Offset,
lxList[inner_end].Length, prefixList, queryData, ServerData);
        newNode.StepEvent += stepEventHandler;
        (newNode.InnerLvl, offs) =
Lexem2Node(lxList, i + 1, inner_end);
        i = inner_end;
        if (rt == null) rt = currentNode = newNode;
        else
        {
            currentNode.SameLvl = newNode;
            currentNode = newNode;
        }
        continue;
    }
    else if (lxList[i].Kind == LexemKind.Exec)
    {
        newNode = new ExecNode(lxList[i].Value,
lxList[i].Offset, lxList[i].Length, prefixList, queryData,
ServerData);

        newNode.StepEvent += stepEventHandler;
        if (rt == null) rt = currentNode = newNode;
        else
        {
            currentNode.SameLvl = newNode;
            currentNode = newNode;
        }
        continue;
    }
    else if (lxList[i].Kind == LexemKind.OAS )
    {
        newNode = new OASNode(lxList[i].Value,
lxList[i].Offset, lxList[i].Length, prefixList, queryData,
ServerData);

        newNode.StepEvent += stepEventHandler;
        if (rt == null) rt = currentNode = newNode;
        else
        {

```

```

        currentNode.SameLvl = newNode;
        currentNode = newNode;
    }
    continue;
}
else if (lxList[i].Kind == LexemKind.OAM)
{
    newNode = new OAMNode(lxList[i].Value,
lxList[i].Offset, lxList[i].Length, prefixList, queryData,
ServerData);

    newNode.StepEvent += stepEventHandler;
    if (rt == null) rt = currentNode = newNode;
    else
    {
        currentNode.SameLvl = newNode;
        currentNode = newNode;
    }
    continue;
}
else if (lxList[i].Kind == LexemKind.OCS )
{
    newNode = new OCSNode(lxList[i].Value,
lxList[i].Offset, lxList[i].Length, prefixList, queryData,
ServerData);

    newNode.StepEvent += stepEventHandler;
    if (rt == null) rt = currentNode = newNode;
    else
    {
        currentNode.SameLvl = newNode;
        currentNode = newNode;
    }
    continue;
}
else if (lxList[i].Kind == LexemKind.PRIOR )
{
    newNode = new PRIORNode(lxList[i].Value,
lxList[i].Offset, lxList[i].Length, prefixList, queryData,
ServerData);

    newNode.StepEvent += stepEventHandler;
    if (rt == null) rt = currentNode = newNode;
    else
    {
        currentNode.SameLvl = newNode;
        currentNode = newNode;
    }
    continue;
}
else if (lxList[i].Kind == LexemKind.RAND)
{
    newNode = new RANDNode(lxList[i].Value,
lxList[i].Offset, lxList[i].Length, prefixList, queryData,
ServerData);

    newNode.StepEvent += stepEventHandler;

```

```

        if (rt == null) rt = currentNode = newNode;
        else
        {
            currentNode.SameLvl = newNode;
            currentNode = newNode;
        }
        continue;
    }
    else
        throw new Exception(string.Format("Wrong
operator: {0}", lxList[i].Offset- lxList[i].Length));
    }
    return (rt, (i - start));
}

private int BlockSearch(List<Lexem> lxList, int start,
int end, LexemKind constrStart, LexemKind constrEnd)
{
    int a = 0;
    do
    {
        if (lxList[start].Kind == constrStart)
            a++;
        if (lxList[start++].Kind == constrEnd)
            a--;
        if (start == end && a != 0)
        {
            throw new
Exception(string.Format("Block Error start from {0}", start));
        }

        } while (a != 0 );
    return start - 1;
}

private (string,string) ConditionRead(string searchStr,
string[] startStrs)
{
    string str = searchStr;
    string condition = string.Empty;
    bool find = false;
    while (!find)
    {
        foreach (string oper in startStrs)
        {
            if (str.StartsWith(oper))
            {
                find = true;
                break;
            }
        }
        if (!find)
        {

```

```

        condition += str[0];
        str = str.Remove(0, 1);
        if (str == string.Empty)
        {
            //error
        }
    }
    //check condition
    return (condition, str);
}

protected virtual void OnMessageEvent(CustomEventArgs e)
{
    MessageEvent?.Invoke(this, e);
}

private void stepEventHandler(object sender,
CustomEventArgs e)
{
    OnMessageEvent(e);
}
}

```

Файл Lexer.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;

namespace SparqlClient
{
    enum LexemKind //добавить программные операторы
    {
        While,
        If,
        Then,
        Else,
        Do,
        End,
        Repeat,
        Until,
        Exec,
        OAS,
        OAM,
        OCS,
        OCM,
        PRIOR,
        RAND,
    }
}

```

```

        Identifier,
        Number
    }
    class LocationEntity
    {
        public int Offset;
        public int Length;
    }
    class Lexem : LocationEntity
    {
        public LexemKind Kind;
        public string Value;
    }
    class LexemDefinition<T>
    {
        public LexemKind Kind { get; protected set; }
        public T Representation { get; protected set; }
    }
    class StaticLexemDefinition : LexemDefinition<string>
    {
        public StaticLexemDefinition(string rep, LexemKind kind)
        {
            Representation = rep;
            Kind = kind;
        }
    }

    static class LexemDefinitions
    {
        public static StaticLexemDefinition[] Statics = new[]
        {
            new StaticLexemDefinition("while", LexemKind.While),
            new StaticLexemDefinition("if", LexemKind.If),
            new StaticLexemDefinition("then", LexemKind.Then),
            new StaticLexemDefinition("else", LexemKind.Else),
            new StaticLexemDefinition("do", LexemKind.Do),
            new StaticLexemDefinition("end", LexemKind.End),
            new StaticLexemDefinition("repeat",
LexemKind.Repeat),
            new StaticLexemDefinition("until", LexemKind.Until),
            new StaticLexemDefinition("exec", LexemKind.Exec),
            new StaticLexemDefinition("OAS", LexemKind.OAS),
            new StaticLexemDefinition("OAM", LexemKind.OAM),
            new StaticLexemDefinition("OCS", LexemKind.OCS),
            new StaticLexemDefinition("PRIOR", LexemKind.PRIOR),
            new StaticLexemDefinition("RAND", LexemKind.RAND),
        };
    }

    class Lexer
    {

```

```

private char[] SpaceChars = new[] { ' ', '\n', '\r',
'\t' };
private string Source;
private int Offset;

public IEnumerable<Lexem> Lexems { get; private set; }

public Lexer(string src)
{
    Source = src;
    Parse();
}

private void Parse()
{
    var lexems = new List<Lexem>();

    while (InBounds())
    {
        SkipSpaces();
        if (!InBounds()) break;

        var lex = ProcessStatic();
        if (lex == null)
            throw new Exception(string.Format("Unknown
lexem at {0}", Offset));

        lexems.Add(lex);
    }

    Lexems = lexems;
}

private bool CheckSpace(char symbol)
{
    foreach (char checkSymbol in SpaceChars)
        if (checkSymbol == symbol)
            return true;
    return false;
}

private void SkipSpaces()
{
    while (InBounds() && CheckSpace(Source[Offset]))
        Offset++;
}

private Lexem ProcessStatic()
{
    string[] startOperators = { "while", "if", "repeat",
"do", "exec" , "OAS", "OAM", "OCS", "PRIOR", "RAND"};
    string[] allOperators = { "while", "if", "else",
"repeat", "exec", "do", "end", "OAS", "OAM", "OCS", "PRIOR",
"RAND" };

```

```

        string condition;
        int conditionOffset;
        int local_offs;
        int length;
        foreach (var def in LexemDefinitions.Statics)
        {
            var rep = def.Representation;
            var len = rep.Length;

            if (Offset + len > Source.Length ||
Source.Substring(Offset, len) != rep)
                continue;

            Offset += len;
            if (def.Kind == LexemKind.While)
            {
                (condition, conditionOffset) =
ConditionRead(Source.Substring(Offset, Source.Length - Offset),
startOperators);
                if (condition == string.Empty)
                    return null;
                local_offs = Offset;
                Offset += conditionOffset;
                length = condition.TrimEnd().Length;
                return new Lexem { Kind = def.Kind, Offset =
local_offs + length, Length = len + length, Value = condition };
            }
            else if (def.Kind == LexemKind.If)
            {
                (condition, conditionOffset) =
ConditionRead(Source.Substring(Offset, Source.Length - Offset),
new string[] { "then" });
                if (condition == string.Empty)
                    return null;
                local_offs = Offset;
                Offset += conditionOffset;
                length = condition.TrimEnd().Length;
                return new Lexem { Kind = def.Kind, Offset =
local_offs + length, Length = len + length, Value = condition };
            }
            else if (def.Kind == LexemKind.Until)
            {
                (condition, conditionOffset) =
ConditionRead(Source.Substring(Offset, Source.Length - Offset),
allOperators, true);
                if (condition == string.Empty)
                    return null;
                local_offs = Offset;
                Offset += conditionOffset;
                length = condition.TrimEnd().Length;
                return new Lexem { Kind = def.Kind, Offset =
local_offs + length, Length = len + length, Value = condition };
            }
        }
    }

```

```

        else if (def.Kind == LexemKind.Exec)
        {
            local_offs = Offset;
            SkipSpaces();
            local_offs = Offset - local_offs;
            string pattern = @"^\w+";
            string rule =
Regex.Match(Source.Substring(Offset, Source.Length - Offset),
pattern).Value;
            Offset += rule.Length;
            return new Lexem { Kind = def.Kind, Offset =
Offset, Length = len + rule.Length + local_offs, Value = rule };
        }
        else if (def.Kind == LexemKind.OAS || def.Kind
== LexemKind.OAM || def.Kind == LexemKind.OCS || def.Kind ==
LexemKind.RAND || def.Kind == LexemKind.PRIOR)
        {
            (condition, conditionOffset) =
BracketRead(Source.Substring(Offset, Source.Length - Offset));
            if (condition == string.Empty)
                return null;
            Offset += conditionOffset;
            return new Lexem { Kind = def.Kind, Offset =
Offset, Length = len + conditionOffset, Value = condition };
        }

        return new Lexem { Kind = def.Kind, Offset =
Offset, Length = len };
    }

    return null;
}

private (string, int) BracketRead(string searchStr)
{
    string str = searchStr;
    string condition = string.Empty;
    int offset = 0;

    if (str[0] == '(')
    {
        offset = 1;
        while (str[offset] != ')')
        {
            condition += str[offset];
            offset++;
        }
        offset++;
    }
    return (condition, offset);
}

```

```

        private (string, int) ConditionRead(string searchStr,
string[] startStrs, bool untilCond = false)
        {
            string str = searchStr;
            string condition = string.Empty;
            bool find = false;
            int offset = 0;
            while (!find)
            {
                foreach (string oper in startStrs)
                {
                    if (offset + oper.Length > str.Length)
                        continue;
                    if (str.Substring(offset, oper.Length) ==
oper)
                    {
                        find = true;
                        break;
                    }
                }
                if (!find)
                {
                    condition += str[offset];
                    offset++;
                    if (offset == str.Length)
                    {
                        if(untilCond)
                            return (condition, offset);
                        return (string.Empty, 0);
                    }
                }
            }

            return (condition, offset);
        }

        private bool InBounds()
        {
            return Offset < Source.Length;
        }
    }
}

```

Файл CodeTree.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net.Http;

```

```

namespace SparqlClient
{
    abstract class CodeTree
    {
        private CodeTree inner;
        private CodeTree same;
        private int offset, length;
        public delegate void StepEventHandler(object sender,
CustomEventArgs e);
        public event StepEventHandler StepEvent;

        public CodeTree(int offs, int len)
        {
            this.offset = offs;
            this.length = len;
            same = null;
            inner = null;
        }

        public abstract Task<bool> Execute();
        public CodeTree InnerLvl
        {
            get { return inner; }
            set { inner = value; }
        }
        public CodeTree SameLvl
        {
            get { return same; }
            set { same = value; }
        }

        public int PositionStart
        {
            get { return offset-length; }
        }
        public int PositionEnd
        {
            get { return offset; }
        }

        protected virtual void OnStepEvent(CustomEventArgs e)
        {
            StepEvent?.Invoke(this, e);
        }

        protected virtual void OnVisualEvent(CustomEventArgs e)
        {
            StepEvent?.Invoke(this, e);
        }
    }
}

```

```

class WhileNode : CodeTree
{
    ConditionParser condition;

    public WhileNode(string str, int offs, int len, string
prefixStr, List<QueryType> queryData, ServerDataType ServerData)
:base(offs,len)
    {
        try
        {
            condition = new ConditionParser(str, prefixStr,
queryData, ServerData);
        }
        catch
        {
            throw new Exception(string.Format("Error in
while-condition: \"{0}\"", str));
        }
    }

    public override async Task<bool> Execute()
    {
        while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
        {
            if (CodeAnalizator.exit)
                return true;
            await Task.Delay(100);
        }

        while (await condition.Result())
        {
            CodeAnalizator.step--;
            OnStepEvent(new
CustomEventArgs(string.Format("WHILE TRUE")));
            await InnerLvl.Execute();
            while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
            {
                if(CodeAnalizator.exit)
                    return true;
                await Task.Delay(100);
            }
        }
        CodeAnalizator.step--;
        OnStepEvent(new CustomEventArgs(string.Format("WHILE
FALSE")));
        if (SameLvl != null)
            await SameLvl.Execute();
        return true;
    }
}

```

```

    }
}

class UntilNode : CodeTree
{
    ConditionParser condition;
    public UntilNode(string str, int offs, int len, string
prefixStr, List<QueryType> queryData, ServerDataType ServerData)
: base(offs, len)
    {
        try
        {
            condition = new ConditionParser(str, prefixStr,
queryData, ServerData);
        }
        catch
        {
            throw new Exception(string.Format("Error in
until-condition: \"{0}\"", str));
        }
    }

    public override async Task<bool> Execute()
    {
        bool send = false;
        bool first = false;
        try
        {
            do
            {
                if (first)
                {
                    CodeAnalizator.step--;
                    OnStepEvent(new
CustomEventArgs(string.Format("UNTIL FALSE")));
                }
                await InnerLvl.Execute();
                first = true;
                while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
                {
                    if (!send)
                    {
                        OnStepEvent(new
CustomEventArgs(PositionStart, PositionEnd));
                        send = true;
                    }
                    if (CodeAnalizator.exit)
                        return true;
                    await Task.Delay(100);
                }
                send = false;
            }
            //ожидание until

```

```

        } while (!(await condition.Result()));
        CodeAnalizator.step--;
        OnStepEvent(new
CustomEventArgs(string.Format("UNTIL TRUE")));
        if (SameLvl != null)
            await SameLvl.Execute();
    }
    catch(Exception e)
    {
        throw new Exception(string.Format(e.Message));
    }
    return true;
}
}
class IfNode : CodeTree
{
    ConditionParser condition;
    public IfNode(string str, int offs, int len, string
prefixStr, List<QueryType> queryData, ServerDataType ServerData)
: base(offs, len)
    {
        try
        {
            condition = new ConditionParser(str, prefixStr,
queryData, ServerData);
        }
        catch
        {
            throw new Exception(string.Format("Error in if-
condition: \"{0}\"",str));
        }
    }

    public override async Task<bool> Execute()
    {
        while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
        {
            if (CodeAnalizator.exit)
                return true;
            await Task.Delay(100);
        }

        if (await condition.Result())
        {
            CodeAnalizator.step--;
            OnStepEvent(new
CustomEventArgs(string.Format("IF TRUE")));
            await InnerLvl.Execute();
        }
        else if (InnerLvl.SameLvl != null)
        {
            CodeAnalizator.step--;

```

```

        OnStepEvent(new
CustomEventArgs(string.Format("IF FALSE")));
        await InnerLvl.SameLvl.Execute();
    }
    else
    {
        CodeAnalizator.step--;
        OnStepEvent(new
CustomEventArgs(string.Format("IF FALSE")));
    }
    if (SameLvl != null)
        await SameLvl.Execute();
    return true;
}

}

class ExecNode : CodeTree
{
    string rule = string.Empty;
    string prefixStr;
    List<QueryType> queryData;
    ServerDataType ServerData;
    public ExecNode(string rule, int offs, int len, string
prefixStr, List<QueryType> queryData, ServerDataType ServerData)
: base(offs, len)
    {
        foreach (QueryType query in queryData)
            if (query.Name == rule)
            {
                this.rule = rule;    //здесь выполнить
проверку правила
                break;
            }
        if (this.rule == string.Empty)
            throw new Exception(string.Format("Error in exec
rule name: \"{0}\" ",rule));

        this.prefixStr = prefixStr;
        this.queryData = queryData;
        this.ServerData = ServerData;
    }

    public override async Task<bool> Execute()
    {
        bool send = false;
        while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
        {
            if (!send)
            {
                OnStepEvent(new
CustomEventArgs(PositionStart, PositionEnd));

```

```

        send = true;
    }
    if (CodeAnalizator.exit)
        return true;
    await Task.Delay(100);
}
send = false;

RuleExecuter sender = new RuleExecuter(rule,
prefixStr, queryData, ServerData);
bool result = await sender.UpdateQuery();
if (result)
{
    OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} COMPLETE", rule)));
}
else
{
    OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} ERROR", rule)));
}
CodeAnalizator.step--;
if (SameLvl != null)
    await SameLvl.Execute();
return true;
}
}

class ThenNode : CodeTree
{
    public ThenNode(int offs, int len) : base(offs, len)
    {
    }

    public override async Task<bool> Execute()
    {
        await InnerLvl.Execute();
        return true;
    }
}

class ElseNode : CodeTree
{
    public ElseNode(int offs, int len) : base(offs, len)
    {
    }

    public override async Task<bool> Execute()
    {
        await InnerLvl.Execute();
        return true;
    }
}

```

```

class OASNode : CodeTree
{
    string[] ruleArray;
    string prefixStr;
    List<QueryType> queryData;
    ServerDataType ServerData;
    public OASNode(string rules, int offs, int len, string
prefixStr, List<QueryType> queryData, ServerDataType ServerData)
: base(offs, len)
    {
        int checkRule = 0;
        ruleArray = rules.Split(',').Select(rule =>
rule.Trim()).ToArray();
        foreach (string rule in ruleArray)
            foreach (QueryType query in queryData)
                if (query.Name == rule)
                {
                    checkRule++;
                    break;
                }
        if (checkRule < ruleArray.Length || ruleArray.Length
== 0)
            throw new Exception(string.Format("Error in OAS
rule list: \"{0}\" ", rules));

        this.prefixStr = prefixStr;
        this.queryData = queryData;
        this.ServerData = ServerData;
    }

    public override async Task<bool> Execute()
    {
        bool send = false;
        while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
        {
            if (!send)
            {
                OnStepEvent(new
CustomEventArgs(PositionStart, PositionEnd));
                send = true;
            }
            if (CodeAnalizator.exit)
                return true;
            await Task.Delay(100);
        }
        send = false;
        foreach (string rule in ruleArray)
        {
            RuleExecuter sender = new RuleExecuter(rule,
prefixStr, queryData, ServerData);

```

```

        bool result = await sender.UpdateQuery();
        if (result)
        {
            OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} COMPLETE", rule)));
        }
        else
        {
            OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} ERROR", rule)));
        }
    }
    CodeAnalizator.step--;
    if (SameLvl != null)
        await SameLvl.Execute();
    return true;
}

}

class OAMNode : CodeTree
{
    string[] ruleArray;
    string prefixStr;
    List<QueryType> queryData;
    ServerDataType ServerData;
    public OAMNode(string rules, int offs, int len, string
prefixStr, List<QueryType> queryData, ServerDataType ServerData)
: base(offs, len)
    {
        int checkRule = 0;
        ruleArray = rules.Split(',').Select(rule =>
rule.Trim()).ToArray();
        foreach (string rule in ruleArray)
            foreach (QueryType query in queryData)
                if (query.Name == rule)
                {
                    checkRule++;
                    break;
                }
        if (checkRule < ruleArray.Length || ruleArray.Length
== 0)
            throw new Exception(string.Format("Error in OAM
rule list: \"{0}\" ", rules));

        this.prefixStr = prefixStr;
        this.queryData = queryData;
        this.ServerData = ServerData;
    }

    public override async Task<bool> Execute()
    {
        bool OrALL = false;

```

```

        bool send = false;
        while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
        {
            if (!send)
            {
                OnStepEvent(new
CustomEventArgs(PositionStart, PositionEnd));
                send = true;
            }
            if (CodeAnalizator.exit)
                return true;
            await Task.Delay(100);
        }
        send = false;
        do
        {
            if (CodeAnalizator.exit)
                return true;
            foreach (string rule in ruleArray)
            {
                RuleExecuter sender = new RuleExecuter(rule,
prefixStr, queryData, ServerData);
                bool result = await sender.UpdateQuery();
                if (result)
                {
                    OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} COMPLETE", rule)));
                }
                else
                {
                    OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} ERROR", rule)));
                }
            }
            foreach (string rule in ruleArray)
            {
                RuleExecuter sender = new RuleExecuter(rule,
prefixStr, queryData, ServerData);
                OrALL |= await sender.AskQuery();
            }
            OnStepEvent(new
CustomEventArgs(string.Format("OR ALL {0}", OrALL)));

        } while (OrALL);
        CodeAnalizator.step--;
        if (SameLvl != null)
            await SameLvl.Execute();
        return true;
    }
}

class OCSNode : CodeTree

```

```

    {
        string[] ruleArray;
        string prefixStr;
        List<QueryType> queryData;
        ServerDataType ServerData;
        public OCSNode(string rules, int offs, int len, string
prefixStr, List<QueryType> queryData, ServerDataType ServerData)
: base(offs, len)
        {
            int checkRule = 0;
            ruleArray = rules.Split(',').Select(rule =>
rule.Trim()).ToArray();
            foreach (string rule in ruleArray)
                foreach (QueryType query in queryData)
                    if (query.Name == rule)
                    {
                        checkRule++;
                        break;
                    }
            if (checkRule < ruleArray.Length || ruleArray.Length
== 0)
                throw new Exception(string.Format("Error in OCS
rule list: \"{0}\" ", rules));

            this.prefixStr = prefixStr;
            this.queryData = queryData;
            this.ServerData = ServerData;
        }

        public override async Task<bool> Execute()
        {
            int loopCntr;
            bool send = false;
            while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
            {
                if (!send)
                {
                    OnStepEvent(new
CustomEventArgs(PositionStart, PositionEnd));
                    send = true;
                }
                if (CodeAnalizator.exit)
                    return true;
                await Task.Delay(100);
            }
            send = false;

            foreach (string rule in ruleArray)
            {
                loopCntr = 0;

```

```

        RuleExecuter sender = new RuleExecuter(rule,
prefixStr, queryData, ServerData);
        while (await sender.AskQuery())
        {
            if (loopCntr > 20)
            {
                OnStepEvent(new
CustomEventArgs(string.Format("LOOP DETECT more 20 time")));
                break;
            }
            bool result = await sender.UpdateQuery();
            if (result)
            {
                OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} COMPLETE", rule)));
            }
            else
            {
                OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} ERROR", rule)));
            }
            loopCntr++;
        }
    }
    CodeAnalizator.step--;
    if (SameLvl != null)
        await SameLvl.Execute();
    return true;
}

}

class PRIORNode : CodeTree
{
    string[] ruleArray;
    string prefixStr;
    List<QueryType> queryData;
    ServerDataType ServerData;
    public PRIORNode(string rules, int offs, int len, string
prefixStr, List<QueryType> queryData, ServerDataType ServerData)
: base(offs, len)
    {
        int checkRule = 0;
        ruleArray = rules.Split(',').Select(rule =>
rule.Trim()).ToArray();
        foreach (string rule in ruleArray)
            foreach (QueryType query in queryData)
                if (query.Name == rule)
                {
                    checkRule++;
                    break;
                }
        if (checkRule < ruleArray.Length || ruleArray.Length
== 0)

```

```

        throw new Exception(string.Format("Error in
PRIOR rule list: \"{0}\" ", rules));

        this.prefixStr = prefixStr;
        this.queryData = queryData;
        this.ServerData = ServerData;
    }

    public override async Task<bool> Execute()
    {
        bool send = false;
        while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
        {
            if (!send)
            {
                OnStepEvent(new
CustomEventArgs(PositionStart, PositionEnd));
                send = true;
            }
            if (CodeAnalizator.exit)
                return true;
            await Task.Delay(100);
        }
        send = false;

        await RepeateRule(ruleArray.ToList<string>());
        if (CodeAnalizator.exit)
            return true;
        CodeAnalizator.step--;
        if (SameLvl != null)
            await SameLvl.Execute();
        return true;
    }

    public async Task<bool> RepeateRule(List<string>
ruleList)
    {
        bool orALL = false;
        List<string> localRules = new List<string>();
        foreach (string rule in ruleList)
            localRules.Add(rule);
        string lastRule = localRules[localRules.Count - 1];
        localRules.RemoveAt(localRules.Count - 1);
        do
        {
            if (CodeAnalizator.exit)
                return true;
            if(localRules.Count>0)
                await RepeateRule(localRules);
            RuleExecuter sender = new RuleExecuter(lastRule,
prefixStr, queryData, ServerData);

```

```

        bool result = await sender.UpdateQuery();
        if (result)
        {
            OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} COMPLETE", lastRule)));
        }
        else
        {
            OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} ERROR", lastRule)));
        }
        foreach (string rule in ruleList)
        {
            RuleExecutor asker = new RuleExecutor(rule,
prefixStr, queryData, ServerData);
            orALL |= await asker.AskQuery();
        }
        } while (orALL);
        return true;
    }
}

class RANDNode : CodeTree
{
    string[] ruleArray;
    string prefixStr;
    List<QueryType> queryData;
    ServerDataType ServerData;
    public RANDNode(string rules, int offs, int len, string
prefixStr, List<QueryType> queryData, ServerDataType ServerData)
: base(offs, len)
    {
        int checkRule = 0;
        ruleArray = rules.Split(',').Select(rule =>
rule.Trim()).ToArray();
        foreach (string rule in ruleArray)
            foreach (QueryType query in queryData)
                if (query.Name == rule)
                {
                    checkRule++;
                    break;
                }
        if (checkRule < ruleArray.Length || ruleArray.Length
== 0)
            throw new Exception(string.Format("Error in RAND
rule list: \"{0}\" ", rules));

        this.prefixStr = prefixStr;
        this.queryData = queryData;
        this.ServerData = ServerData;
    }
}

```

```

public override async Task<bool> Execute()
{
    bool OrALL = false;
    bool send = false;
    Random rnd = new Random();
    while (CodeAnalizator.step == 0 &&
!CodeAnalizator.run)
    {
        if (!send)
        {
            OnStepEvent(new
CustomEventArgs(PositionStart, PositionEnd));
            send = true;
        }
        if (CodeAnalizator.exit)
            return true;
        await Task.Delay(100);
    }
    send = false;
    do
    {
        if (CodeAnalizator.exit)
            return true;
        int execRule = rnd.Next(0, ruleArray.Length);

        RuleExecuter sender = new
RuleExecuter(ruleArray[execRule], prefixStr, queryData,
ServerData);
        bool result = await sender.UpdateQuery();
        if (result)
        {
            OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} COMPLETE",
ruleArray[execRule])));
        }
        else
        {
            OnStepEvent(new
CustomEventArgs(string.Format("EXEC {0} ERROR",
ruleArray[execRule])));
        }

        foreach (string rule in ruleArray)
        {
            RuleExecuter asker = new RuleExecuter(rule,
prefixStr, queryData, ServerData);
            OrALL |= await asker.AskQuery();
        }
        OnStepEvent(new
CustomEventArgs(string.Format("OR ALL {0}", OrALL)));
    } while (OrALL);

    CodeAnalizator.step--;
}

```

```

        if (SameLvl != null)
            await SameLvl.Execute();
        return true;
    }
}

```

Файл ConditionParser.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SparqlClient
{
    class ConditionParser
    {
        Dictionary<string, Func<Node>> functions;
        Node root;

        //public string error;

        string prefixList;
        List<QueryType> queryData;
        ServerDataType ServerData;
        List<string> rules;

        public ConditionParser(string str, string prefixStr,
            List<QueryType> queryData, ServerDataType ServerData)
        {
            this.prefixList = prefixStr;
            this.queryData = queryData;
            this.ServerData = ServerData;
            rules = new List<string>();
            foreach (QueryType query in queryData)
            {
                rules.Add(query.Name);
            }

            functions = new Dictionary<string, Func<Node>>
            {
                { "A", () => new QueryA(prefixStr,
                    queryData, ServerData) },
                { "C", () => new QueryC(prefixStr,
                    queryData, ServerData) }
            };

            str = str.Trim();
            root = Expression(str, rules.ToArray());
        }
    }
}

```

```

public async Task<bool> Result()
{
    return await root.ValueBool();
}

Node Expression(string strTemp, string[] rules)
{
    strTemp = strTemp.Trim();
    try
    {
        int i = 0;
        while (i < strTemp.Length)
        {
            if (strTemp[i] == '(')
            {
                i = Brackets(strTemp, i);
                continue;
            }
            if ((strTemp.Length - i) >= 2)
                if (strTemp.Substring(i,
2).StartsWith("or"))
                {
                    if (strTemp.Substring(0, i) == "")
                    {
                        i++;
                        continue;
                    }
                    Node rt = new Or();
                    rt.Left =
Expression(strTemp.Substring(0, i - 1), rules);
                    rt.Rigth =
Expression(strTemp.Substring(i + 2, strTemp.Length - (i + 2)),
rules);
                    return rt;
                }
                i++;
            }

            return Expression1(strTemp, rules);
        }
        catch
        {
            throw new Exception("Введена некорректная
строка");
        }
    }

    Node Expression1(string strTemp, string[] rules)
    {
        try
        {
            int i = 0;
            while (i < strTemp.Length)

```

```

        {
            if (strTemp[i] == '(')
            {
                i = Brackets(strTemp, i);
                continue;
            }
            if ((strTemp.Length - i) >= 3)
                if (strTemp.Substring(i,
3).StartsWith("and"))
                {
                    if (strTemp.Substring(0, i) == "")
                    {
                        i++;
                        continue;
                    }
                    Node rt = new And();
                    rt.Left =
Expression(strTemp.Substring(0, i - 1), rules);
                    rt.Rigth =
Expression(strTemp.Substring(i + 3, strTemp.Length - (i + 3)),
rules);
                    return rt;
                }
                i++;
            }

            return Term(strTemp, rules);
        }
    catch
    {
        throw new Exception("Wrong condition string");
    }
}

Node Term(string strTemp, string[] rules)
{
    try
    {
        int i = 0, p = -1;
        while (i < strTemp.Length)
        {
            if (strTemp[i] == '(')
            {
                i = Brackets(strTemp, i);
                continue;
            }
            if ((strTemp.Length - i) >= 3)
                if (strTemp.Substring(i,
3).StartsWith("not"))
                {
                    Node rt = new Not();

```

```

        rt.Left =
Expression(strTemp.Substring(i + 4, strTemp.Length - (i + 4)),
rules);

        return rt;
    }
    i++;
}
return Term1(strTemp, rules);
}
catch
{
    throw new Exception("Wrong condition string");
}
}

Node Term1(string strTemp, string[] rules)
{
    try
    {
        int i = 0, p = -1;
        while (i < strTemp.Length)
        {
            if (strTemp[i] == '(')
            {
                i = Brackets(strTemp, i);
                continue;
            }
            if (strTemp[i] == '>' || strTemp[i] == '<'
|| strTemp[i] == '=')
            {
                if (strTemp.Substring(0, i) == "")
                {
                    i++;
                    continue;
                }
                Node rt = new Check(strTemp[i]);
                rt.Left =
Expression(strTemp.Substring(0, i), rules);
                rt.Rigth =
Expression(strTemp.Substring(i + 1, strTemp.Length - (i + 1)),
rules);

                return rt;
            }
            i++;
        }
        return Factor(strTemp, rules);
    }
    catch
    {
        throw new Exception("Wrong condition string");
    }
}
}

```

```

Node Factor(string strTemp, string[] rules)
{
    try
    {
        if (strTemp.StartsWith("A") ||
strTemp.StartsWith("C"))
        {
            if ((strTemp.Length >= 1) &&
(functions.ContainsKey(strTemp.Substring(0, 1)))
            {
                Node rt = functions[strTemp.Substring(0,
1)]();
                rt.Left = Factor(strTemp.Substring(1,
strTemp.Length - (1)), rules);
                return rt;
            }
        }
        if (strTemp[0] == '(' && strTemp[strTemp.Length
- 1] == ')')
        {
            foreach (string rule in rules)
                if (strTemp.Substring(1, strTemp.Length
- 1).StartsWith(rule))
                {
                    strTemp = strTemp.Remove(0, 1);
                    Node rt = new Rule(rule);
                    return rt;
                }
            strTemp = strTemp.Remove(0, 1);
            return Expression(strTemp =
strTemp.Remove(strTemp.Length - 1), rules);
        }
        Node node = new Const(Int32.Parse(strTemp));
        return node;
    }
    catch
    {
        throw new Exception("Wrong condition string");
    }
}

int Brackets(string strTemp, int i)
{
    try
    {
        int a = 0;
        do
        {
            if (strTemp[i] == '(')
                a++;
            if (strTemp[i++] == ')')

```

```

        a--;
    } while (a != 0);
    return i;
}
catch
{
    throw new Exception("Wrong condition string");
}
}

abstract class Node
{
    protected Node left;
    protected Node righth;

    public Node Left
    {
        set
        {
            left = value;
        }
    }
    public Node Righth
    {
        set
        {
            righth = value;
        }
    }
    public virtual async Task<bool> ValueBool()
    {
        return false;
    }
    public virtual async Task<int> ValueInt()
    {
        return 0;
    }
    public virtual string ValueString()
    {
        return string.Empty;
    }
}

class Or : Node
{
    public override async Task<bool> ValueBool()
    {
        return await left.ValueBool() || await
righth.ValueBool();
    }
}

```

```

class And : Node
{
    public override async Task<bool> ValueBool()
    {
        return await left.ValueBool() && await
righth.ValueBool();
    }
}

class Check : Node
{
    char op;
    public Check(char op)
    {
        this.op = op;
    }

    public override async Task<bool> ValueBool()
    {
        if (op == '>')
            return await left.ValueInt() > await
righth.ValueInt();
        else if (op == '<')
            return await left.ValueInt() < await
righth.ValueInt();
        else
            return await left.ValueInt() == await
righth.ValueInt();
    }
}

class Rule : Node
{
    string rule;
    public Rule(string rule)
    {
        this.rule = rule;
    }

    public override string ValueString()
    {
        //составление текста правила
        return rule;
    }
}

class Not : Node
{
    public override async Task<bool> ValueBool()
    {
        return !(await left.ValueBool());
    }
}

class QueryA : Node

```

```

{
    string prefixList;
    List<QueryType> queryData;
    ServerDataType ServerData;
    public QueryA(string prefixStr, List<QueryType>
queryData, ServerDataType ServerData)
    {
        this.prefixList = prefixStr;
        this.queryData = queryData;
        this.ServerData = ServerData;
    }

    public override async Task<bool> ValueBool()
    {
        string ruleName = left.ValueString();
        RuleExecuter sender = new RuleExecuter(ruleName,
prefixList, queryData, ServerData);
        return await sender.AskQuery();
    }
}

class QueryC : Node
{
    string prefixList;
    List<QueryType> queryData;
    ServerDataType ServerData;
    public QueryC(string prefixStr, List<QueryType>
queryData, ServerDataType ServerData)
    {
        this.prefixList = prefixStr;
        this.queryData = queryData;
        this.ServerData = ServerData;
    }
    public override async Task<int> ValueInt()
    {
        string ruleName = left.ValueString();
        RuleExecuter sender = new RuleExecuter(ruleName,
prefixList, queryData, ServerData);
        return await sender.CountQuery();
    }
}

class Const : Node
{
    int a;
    public Const(int a)
    {
        this.a = a;
    }

    public override async Task<int> ValueInt()
    {
        return a;
    }
}

```

```

    }
}

```

Файл RuleExecuter.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net.Http;

namespace SparqlClient
{
    class RuleExecuter
    {
        HttpClient client = new HttpClient();
        string prefixList;
        List<QueryType> queryData;
        ServerDataType ServerData;

        QueryType rule;

        public RuleExecuter(string ruleName, string prefixStr,
List<QueryType> queryData, ServerDataType ServerData)
        {
            this.prefixList = prefixStr;
            this.queryData = queryData;
            this.ServerData = ServerData;

            foreach (QueryType query in queryData)
            {
                if (query.Name == ruleName)
                {
                    rule = query;
                    break;
                }
            }
        }

        private string GetWhereQuery(string query)
        {
            string result = string.Empty;
            string[] words = query.Split(new string[] { "\r\n"
}, StringSplitOptions.RemoveEmptyEntries);
            bool flagRead = false;
            bool flagCheack = true;
            foreach (string word in words)
            {
                if (flagCheack)
                    if (word.StartsWith("WHERE") ||
word.StartsWith("where"))

```

```

        {
            flagRead = true;
            flagCheack = false;
        }
        if (flagRead)
        {
            result += word + "\r\n";
        }
    }
    return result;
}

public async Task<bool> UpdateQuery()
{
    string queryStr = prefixList + rule.Data;
    StringContent cont = new StringContent("update=" +
queryStr.Replace("+", "%2B"), Encoding.UTF8, "application/x-www-
form-urlencoded");
    HttpResponseMessage response = await
client.PostAsync("http://" + ServerData.IpAddress + ":" +
ServerData.Port + "/" + ServerData.DatasetName + "/update",
cont);
    var jsonResponse = await
response.Content.ReadAsStringAsync();
    string result = jsonResponse.ToString();
    ServerResultParser worker = new
ServerResultParser();
    string parse = worker.ParseUpdate(result);
    if (parse.StartsWith("Success"))
    {
        return true;
    }
    else
    {
        return false;
    }
}

public async Task<bool> AskQuery()
{
    string queryStr = prefixList + " ASK " +
GetWhereQuery(rule.Data);
    StringContent cont = new StringContent(queryStr,
Encoding.UTF8, "application/sparql-query");
    HttpResponseMessage response = await
client.PostAsync("http://" + ServerData.IpAddress + ":" +
ServerData.Port + "/" + ServerData.DatasetName + "/sparql",
cont);
    var jsonResponse = await
response.Content.ReadAsStringAsync();
    string result = jsonResponse.ToString();

```

```

        ServerResultParser worker = new
ServerResultParser();
        string boolRes = worker.ParseBool(result);
        if (boolRes.StartsWith("true"))
        {
            return true;
        }
        else if (boolRes.StartsWith("false"))
        {
            return false;
        }
        else
        {
            throw new Exception(string.Format("ASK " +
rule.Name + ": Parse response error"));
        }
    }

    public async Task<int> CountQuery()
    {
        string queryStr = prefixList + " SELECT (COUNT(*) AS
?n) " + GetWhereQuery(rule.Data);
        StringContent cont = new StringContent(queryStr,
Encoding.UTF8, "application/sparql-query");
        HttpResponseMessage response = await
client.PostAsync("http://" + ServerData.IpAddress + ":" +
ServerData.Port + "/" + ServerData.DatasetName + "/sparql",
cont);

        var jsonResponse = await
response.Content.ReadAsStringAsync();
        string result = jsonResponse.ToString();
        ServerResultParser worker = new
ServerResultParser();
        return int.Parse(worker.ParseCount(result));
    }
}
}

```

Файл ServerResultParser.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using System.Drawing;
using System.Windows.Forms;
using System.Data;

namespace SparqlClient
{
    class ServerResultParser

```

```

    {
        public void ParseXml(string XmlStr, DataGridView
XmlDataGridView)
        {
            try
            {
                XmlDocument xmlDoc = new XmlDocument();
                xmlDoc.LoadXml(XmlStr);

                DataTable dt = new DataTable();

                dt.Columns.Add("#", typeof(string));

                // Чтение заголовков из head
                XmlNodeList headNodes =
xmlDoc.GetElementsByTagName("head");
                foreach (XmlNode headNode in headNodes)
                {
                    foreach (XmlNode variableNode in
headNode.ChildNodes)
                    {
                        string columnName =
variableNode.Attributes["name"].Value;
                        dt.Columns.Add(columnName,
typeof(string));
                    }
                }
                // Чтение данных из XML и добавление их в
DataTable

                XmlNodeList results =
xmlDoc.GetElementsByTagName("result");
                int index = 1;
                foreach (XmlNode result in results)
                {
                    DataRow row = dt.NewRow();
                    row["#"] = index++;
                    foreach (XmlNode binding in
result.ChildNodes)
                    {
                        string name =
binding.Attributes["name"].Value;
                        string value = binding.ChildNodes.Count
> 0 ? ExtractFragment(binding.ChildNodes[0].InnerText) :
string.Empty;
                        row[name] = value;
                    }

                    dt.Rows.Add(row);
                }

                // Привязка DataTable к GridView
                XmlDataGridView.DataSource = dt;
                XmlDataGridView.AutoGenerateColumns = true;
            }
        }
    }

```

```

        }
        catch (Exception ex)
        {
            MessageBox.Show("Error loading XML file:
{ex.Message}", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    static string ExtractFragment(string uri)
    {
        int index = uri.LastIndexOf('#');
        return (index != -1 && index < uri.Length - 1) ?
uri.Substring(index + 1) : uri;
    }

    public string ParseBool(string XmlStr)
    {
        try
        {
            XmlDocument xmlDoc = new XmlDocument();
            xmlDoc.LoadXml(XmlStr);
            XmlNodeList node =
xmlDoc.GetElementsByTagName("boolean");
            return node[0].InnerText;
        }
        catch (Exception ex)
        {
        }
        return "null";
    }

    public string ParseCount(string XmlStr)
    {
        try
        {
            XmlDocument xmlDoc = new XmlDocument();
            xmlDoc.LoadXml(XmlStr);
            XmlNodeList results =
xmlDoc.GetElementsByTagName("result");
            XmlNode binding = results[0].ChildNodes[0];
            string value = binding.ChildNodes.Count > 0 ?
ExtractFragment(binding.ChildNodes[0].InnerText) : string.Empty;
            return value;
        }
        catch (Exception ex)
        {
        }
        return "null";
    }
}

```

```

        public string ParseUpdate(string XmlStr)
        {
            try
            {
                XmlDocument xmlDoc = new XmlDocument();
                xmlDoc.LoadXml(XmlStr);
                XmlNodeList results =
xmlDoc.GetElementsByTagName("body");
                XmlNode binding = results[0].ChildNodes[0];
                string value = binding.ChildNodes.Count > 0 ?
ExtractFragment(binding.ChildNodes[0].InnerText) : string.Empty;
                return value;
            }
            catch (Exception ex)
            {
                return "error";
            }
        }
    }
}

```

Файл CustomEventArgs.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SparqlClient
{
    public class CustomEventArgs : EventArgs
    {
        public string Message { get; }
        public int PositionStart { get; }
        public int PositionEnd { get; }
        public CustomEventArgs(string message)
        {
            Message = message;
        }
        public CustomEventArgs(int start, int end)
        {
            Message = "visual";
            PositionStart = start;
            PositionEnd = end;
        }
    }
}

```

Файл RDFManager.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;

```

```

using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text.Json;
using System.Text.Json.Nodes;
using System.IO;

namespace SparqlClient
{
    public partial class RDFManager : UserControl
    {
        HttpClient client = new HttpClient();
        public RDFManager()
        {
            InitializeComponent();
        }

        static async Task<string> SendShowListRequest(HttpClient
handler)
        {
            HttpRequestMessage requestMessage = new
HttpRequestMessage(HttpMethod.Get,
"http://127.0.0.1:3030/$/server");
            requestMessage.Headers.Add("Accept",
"application/json");
            HttpResponseMessage response = await
handler.SendAsync(requestMessage);
            var jsonResponse = await
response.Content.ReadAsStringAsync();
            // box.Text = jsonResponse.ToString();

            JsonNode forecastNode =
JsonNode.Parse(jsonResponse.ToString());
            JsonArray datasetsNode =
forecastNode["datasets"].AsArray();
            string result = string.Empty;
            //box.Text = "Active datasets:\n";
            foreach (JsonNode node in datasetsNode)
            {
                result += "\t"+node["ds.name"].ToString()+"\n";
            }
            return result;
        }

        static async Task SendCreateRequest(HttpClient handler,
string name)

```

```

        {
            HttpRequestMessage requestMessage = new
HttpRequestMessage (HttpMethod.Post,
"http://127.0.0.1:3030/$/datasets");
            requestMessage.Content = new
StringContent ("dbName="+name+"&dbType=mem", Encoding.UTF8,
"application/x-www-form-urlencoded");
            HttpResponseMessage response = await
handler.SendAsync(requestMessage);
            await response.Content.ReadAsStringAsync();
        }

        static async Task SendShowRequest(HttpClient handler,
RichTextBox box, string name)
        {
            HttpRequestMessage requestMessage = new
HttpRequestMessage (HttpMethod.Get,
"http://127.0.0.1:3030/"+name+"?graph=default");
            requestMessage.Headers.Add("Accept", "text/turtle;
charset=utf-8");
            HttpResponseMessage response = await
handler.SendAsync(requestMessage);
            var turtleResponse = await
response.Content.ReadAsStringAsync();
            box.Text = "Dataset data:\n\n";
            box.Text += turtleResponse.ToString();
        }

        static async Task SendDeleteRequest(HttpClient handler,
string name)
        {
            HttpRequestMessage requestMessage = new
HttpRequestMessage (HttpMethod.Delete,
"http://127.0.0.1:3030/$/datasets/"+name);
            HttpResponseMessage response = await
handler.SendAsync(requestMessage);
            var Response = await
response.Content.ReadAsStringAsync();
        }

        static async Task SendUploadRequest(HttpClient handler,
string datasetName, string filePath)
        {
            string fileName =
Path.GetFileNameWithoutExtension(filePath);
            var multipartFormContent = new
MultipartFormDataContent();
            // Загружаем отправляемый файл
            var fileStreamContent = new
StreamContent (File.OpenRead(filePath));
            multipartFormContent.Add(fileStreamContent, name:
"file", fileName: fileName);

```

```

        fileStreamContent.Headers.ContentType = new
MediaTypeHeaderValue("application/octet-stream");

        HttpRequestMessage requestMessage = new
HttpRequestMessage(HttpMethod.Post, "http://127.0.0.1:3030/"+
datasetName);
        requestMessage.Content = multipartFormContent;
        HttpResponseMessage response = await
handler.SendAsync(requestMessage);
    }

    private async void show_dataset_Click(object sender,
EventArgs e)
    {
        string datasetName;
        EnterValueForm dialog = new EnterValueForm("Enter
dataset name");
        DialogResult result = dialog.ShowDialog();
        if (result == DialogResult.OK)
        {
            datasetName = dialog.Value;
            await SendShowRequest(client, RDFView,
datasetName);
        }
    }

    private async void Show_dataset_list_Click(object
sender, EventArgs e)
    {
        try {
            string resp = await SendShowListRequest(client);
            RDFView.Text = "Active datasets:\n" + resp;
        }
        catch
        {
            RDFView.Text = "No answer from server";
        }
    }

    private async void create_dataset_Click(object sender,
EventArgs e)
    {
        string datasetName;
        EnterValueForm dialog = new EnterValueForm("Enter
dataset name");
        DialogResult result = dialog.ShowDialog();
        if (result == DialogResult.OK)
        {
            datasetName = dialog.Value;
            await SendCreateRequest(client, datasetName);
            string resp = await SendShowListRequest(client);
            RDFView.Text = "Active datasets:\n" + resp;
        }
    }

```

```

        }
    }

    private async void upload_dataset_Click(object sender,
EventArgs e)
    {
        string datsetName;
        EnterValueForm dialog = new EnterValueForm("Enter
dataset name");
        DialogResult result = dialog.ShowDialog();
        if (result == DialogResult.OK)
        {
            datsetName = dialog.Value;
            using (OpenFileDialog openFileDialog = new
OpenFileDialog())
            {
                openFileDialog.InitialDirectory = "c:\\";
                openFileDialog.Filter = "owl files
(*.owl)|*.owl|All files (*.*)|*.*";
                openFileDialog.FilterIndex = 2;
                openFileDialog.RestoreDirectory = true;

                if (openFileDialog.ShowDialog() ==
DialogResult.OK)
                {
                    var filePath = openFileDialog.FileName;
                    await SendUploadRequest(client,
datsetName, filePath);
                }
            }
        }
    }

    private async void delete_dataset_Click(object sender,
EventArgs e)
    {
        string datsetName;
        EnterValueForm dialog = new EnterValueForm();
        DialogResult result = dialog.ShowDialog();
        if (result == DialogResult.OK)
        {
            datsetName = dialog.Value;
            await SendDeleteRequest(client, datsetName);
            string resp = await SendShowListRequest(client);
            RDFView.Text = "Active datasets:\n" + resp;
        }
    }
}

```

Файл EnterValueForm.cs

```

using System;
using System.Collections.Generic;

```

```

using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Text.RegularExpressions;

namespace SparqlClient
{
    public partial class EnterValueForm : Form
    {
        public string value;

        public string Value
        {
            get { return value; }
        }
        public EnterValueForm()
        {
            InitializeComponent();
            ErrorLabel.Text = "";
        }
        public EnterValueForm(string name)
        {
            InitializeComponent();
            ErrorLabel.Text = "";
            this.Text = name;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Regex regExp = new Regex("[^A-Za-z0-9]");

            value = textBox.Text;

            if (value == string.Empty)
                ErrorLabel.Text = "Empty input field ";
            else
            {
                Match m = regExp.Match(value);
                if (m.Success)
                {
                    ErrorLabel.Text = "Incorrect value";
                }
                else
                {
                    DialogResult = DialogResult.OK;
                }
            }
        }
    }
}

```

```

        this.Close();
    }
}
}
}
}

```

Файл QueryEditorForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
namespace SparqlClient
{
    public partial class QueryEditorForm : Form//UserControl
    {
        string prefixList;
        List<QueryType> queryData;

        public QueryEditorForm(string prefixList,
List<QueryType> queryData)
        {
            InitializeComponent();
            this.prefixList = prefixList;
            this.queryData = queryData;
            queryListBox.DataSource = this.queryData;
            queryListBox.DisplayMember = "Name";
            queryListBox.SelectedIndexChanged +=
queryListBox_SelectedIndexChanged;
            nameBox.Text = string.Empty;
            commentBox.Text = string.Empty;
            editorBox.Text = string.Empty;
            saveEditButton.Enabled = false;
        }

        public string PrefixList
        {
            get
            {
                return prefixList;
            }
        }
        public List<QueryType> QueryData
        {
            get
            {
                return queryData;
            }
        }
    }
}

```

```

        }
    }

    void queryListBox_SelectedIndexChanged(object sender,
EventArgs e)
    {
        QueryType query =
(QueryType)queryListBox.SelectedValue;
        nameBox.Text = query.Name;
        commentBox.Text = query.Comment;
        editorBox.Text = query.Data;
        saveEditButton.Enabled = true;
    }

    private void prefixEditButton_Click(object sender,
EventArgs e)
    {
        PrefixForm dialog = new PrefixForm(prefixList);
        dialog.ShowDialog();
        prefixList = dialog.prefixList;
    }

    private void newButton_Click(object sender, EventArgs e)
    {
        queryListBox.SelectedIndexChanged -=
queryListBox_SelectedIndexChanged;
        QueryType query = new QueryType();
        int count = 0;
        while (queryData.Any(q => q.Name ==
"NEW_QUERY"+count))
        {
            count++;
        }
        query.Name = "NEW_QUERY"+count;
        query.Data = "DELETE {\n\n}\nINSERT {\n\n}\nWHERE
{\nSELECT\nWHERE {\n\n}\nLIMIT 1\n}\n";
        queryData.Add(query);
        queryListBox.DataSource = null;
        queryListBox.DataSource = queryData;
        queryListBox.DisplayMember = "Name";
        queryListBox.SelectedIndexChanged +=
queryListBox_SelectedIndexChanged;

        queryListBox.SetSelected(queryData.Count - 1, true);
    }

    private void deleteButton_Click(object sender, EventArgs
e)
    {
        queryListBox.SelectedIndexChanged -=
queryListBox_SelectedIndexChanged;
        // The Remove button was clicked.
        int selectedIndex = queryListBox.SelectedIndex;

```

```

        try
        {
            // Remove the item in the List.
            queryData.RemoveAt(selectedIndex);
        }
        catch
        {
        }
        queryListBox.DataSource = null;
        queryListBox.DataSource = queryData;
        queryListBox.DisplayMember = "Name";
        queryListBox.SelectedIndexChanged +=
queryListBox_SelectedIndexChanged;

        nameBox.Text = string.Empty;
        commentBox.Text = string.Empty;
        editorBox.Text = string.Empty;
        saveEditButton.Enabled = false;
    }

    private void saveListButton_Click(object sender,
EventArgs e)
    {
        using (SaveFileDialog saveFileDialog = new
SaveFileDialog())
        {
            saveFileDialog.Filter = "Text files(*.txt) |
*.txt |All files(*.*)|*.*";
            if (saveFileDialog.ShowDialog() ==
DialogResult.Cancel)
                return;
            string filename = saveFileDialog.FileName;
            string saveFile = "#PREFIX\n" + prefixList;
            foreach (QueryType query in queryData)
                saveFile += "#QUERY " + query.Name + " " +
query.Comment + "\n"+query.Data+"\n";
            saveFile += "#END\n";
            System.IO.File.WriteAllText(filename, saveFile);
        }
    }

    private void saveEditButton_Click(object sender,
EventArgs e)
    {
        queryListBox.SelectedIndexChanged -=
queryListBox_SelectedIndexChanged;
        int selectedIndex = queryListBox.SelectedIndex;
        queryData[selectedIndex].Name = nameBox.Text;
        queryData[selectedIndex].Comment = commentBox.Text;
        queryData[selectedIndex].Data = editorBox.Text;
        queryListBox.DataSource = null;
    }

```

```

        queryListBox.DataSource = queryData;
        queryListBox.DisplayMember = "Name";
        queryListBox.SelectedIndexChanged +=
queryListBox_SelectedIndexChanged;
    }
}
}

```

Файл PrefixForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SparqlClient
{
    public partial class PrefixForm : Form
    {
        string rdf = "PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syax-ns#>";
        string rdfs = "PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>";
        string owl = "PREFIX owl:
<http://www.w3.org/2002/07/owl#>";
        string xsd = "PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>";

        public string prefixList;

        public PrefixForm(string prefixList)
        {
            InitializeComponent();
            prefixBox.Text = prefixList;
            this.prefixList = prefixList;
            if (prefixBox.Text.Contains(rdf))
rdfButton.BackColor = Color.Yellow; else rdfButton.BackColor =
Color.LightGray;
            if (prefixBox.Text.Contains(rdfs))
rdfsButton.BackColor = Color.Yellow; else rdfsButton.BackColor =
Color.LightGray;
            if (prefixBox.Text.Contains(owl))
owlButton.BackColor = Color.Yellow; else owlButton.BackColor =
Color.LightGray;
            if (prefixBox.Text.Contains(xsd))
xsdButton.BackColor = Color.Yellow; else xsdButton.BackColor =
Color.LightGray;
        }
    }
}

```

```

private void rdfButton_Click(object sender, EventArgs e)
{
    if (prefixBox.Text.Contains(rdf))
    {
        string[] str = prefixBox.Text.Split(new string[]
{ rdf, "\n", "\r" },
System.StringSplitOptions.RemoveEmptyEntries);
        prefixBox.Text = string.Empty;
        foreach (string s in str)
            prefixBox.Text += s + "\r\n";
        rdfButton.BackColor = Color.LightGray;
    }
    else
    {
        prefixBox.Text += rdf + "\r\n";
        rdfButton.BackColor = Color.Yellow;
    }
}

private void rdfsButton_Click(object sender, EventArgs
e)
{
    if (prefixBox.Text.Contains(rdfs))
    {
        string[] str = prefixBox.Text.Split(new string[]
{ rdfs, "\n", "\r" },
System.StringSplitOptions.RemoveEmptyEntries);
        prefixBox.Text = string.Empty;
        foreach (string s in str)
            prefixBox.Text += s + "\r\n";
        rdfsButton.BackColor = Color.LightGray;
    }
    else
    {
        prefixBox.Text += rdfs + "\r\n";
        rdfsButton.BackColor = Color.Yellow;
    }
}

private void owlButton_Click(object sender, EventArgs e)
{
    if (prefixBox.Text.Contains(owl))
    {
        string[] str = prefixBox.Text.Split(new string[]
{ owl, "\n", "\r" },
System.StringSplitOptions.RemoveEmptyEntries);
        prefixBox.Text = string.Empty;
        foreach (string s in str)
            prefixBox.Text += s + "\r\n";
        owlButton.BackColor = Color.LightGray;
    }
    else

```

```

        {
            prefixBox.Text += owl + "\r\n";
            owlButton.BackColor = Color.Yellow;
        }
    }

    private void xsdButton_Click(object sender, EventArgs e)
    {
        if (prefixBox.Text.Contains(xsd))
        {
            string[] str = prefixBox.Text.Split(new string[]
{ xsd, "\n", "\r" },
System.StringSplitOptions.RemoveEmptyEntries);
            prefixBox.Text = string.Empty;
            foreach (string s in str)
                prefixBox.Text += s + "\r\n";
            xsdButton.BackColor = Color.LightGray;
        }
        else
        {
            prefixBox.Text += xsd + "\r\n";
            xsdButton.BackColor = Color.Yellow;
        }
    }

    private void saveButton_Click(object sender, EventArgs
e)
    {
        prefixList = prefixBox.Text;
    }
}

```

Файл HelpForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SparqlClient
{
    public partial class HelpForm : Form
    {
        public HelpForm()
        {
            InitializeComponent();
            listBox.SelectedIndexChanged +=
listBox_SelectedIndexChanged;

```

```

    }

    void listBox_SelectedIndexChanged(object sender,
EventArgs e)
    {
        int index = listBox.SelectedIndex;
        textBox.Text = text[index];
    }

    string[] text = {"Getting started with SPARQL/C-A:\n"+
        "1) Open \"Options\" window and setting
SPARQL server ip, port and dataset name\n" +
        "2) Load SPARQL UPDATE rules File-
>Open\n" +
        "That's all you are ready for work",
        "Automatic control: This panel allows
you to load, save and execute"+
        "custom rule execution models in
step-by-step and automatic modes."+
        "The model execution steps will be
displayed in a message window. Errors"+
        "in the model program will also be
displayed in the message window",
        "RDF manager:\n" +
        "This interface allows the user to
manipulate SPARQL server datasets.\n",
        "Query:\n"+
        "This interface allows the user to
execute and test SPARQL queries."+
        "The upper field is intended for
entering a query, the lower one is for displaying the result in
the form of" +
        " a table or XML document. The execute
button sends SPARQL query to the server.",
        "Authors:\n student Ruchkin M.A.\n
student Dubinin A.V.\n D.Eng Dubinin V.N.\nPenza State
University\n 2024"};

    }
}

```

ПРИЛОЖЕНИЕ Б

(Обязательное)

Реализация распределенной графо-трансформационной

системы на основе онтологий и языка SPARQL

Онтология задачи поиска кратчайших путей

```

<?xml version="1.0"?>
<rdf:RDF
xmlns="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#"

xml:base="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15"/>
    <!--

////////////////////////////////////
////////////////////////////////////
    //
    // Object Properties
    //

////////////////////////////////////
////////////////////////////////////
    -->
    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#comes_from -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#comes_from"/>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#comes_to -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#comes_to"/>

    <!--

////////////////////////////////////
////////////////////////////////////
    //
    // Data properties
    //

////////////////////////////////////
////////////////////////////////////
    -->

```

```

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#has_attr_d_value -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#has_attr_d_value">
        <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
    </owl:DatatypeProperty>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#has_attr_max_value -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#has_attr_max_value">
        <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
    </owl:DatatypeProperty>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#has_attr_n_value -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#has_attr_n_value">
        <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#has_color -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#has_color">
        <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#has_edge_type -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#has_edge_type">

```

```

        <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#has_name -->

        <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#has_name">
            <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>

        <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#has_node_type -->

        <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#has_node_type">
            <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>

        <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#has_shape -->

        <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#has_shape">
            <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>

        <!--

////////////////////////////////////
////////////////////////////////////
//
// Classes
//

////////////////////////////////////
////////////////////////////////////
-->

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#Edge -->

```

```

    <owl:Class
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#Edge">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#comes_from"/>
                <owl:qualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeIntege
r">1</owl:qualifiedCardinality>
                    <owl:onClass
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Node"/>
                        </owl:Restriction>
                    </rdfs:subClassOf>
                <rdfs:subClassOf>
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#comes_to"/>
                            <owl:qualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeIntege
r">1</owl:qualifiedCardinality>
                                <owl:onClass
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Node"/>
                                    </owl:Restriction>
                                </rdfs:subClassOf>
                            </owl:Class>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#Node -->

    <owl:Class
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#Node"/>

    <!--

////////////////////////////////////
////////////////////////////////////
//
// Individuals
//

////////////////////////////////////
////////////////////////////////////
-->

```

```

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_ab -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_ab">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
            <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_a"/>
                <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_b"/>
                    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_a
ttr_d_value>
                        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</ha
s_edge_type>
                    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_ae -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_ae">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
            <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_a"/>
                <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_e"/>
                    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_a
ttr_d_value>
                        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</ha
s_edge_type>
                    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_ag -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_ag">
    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Edge"/>
    <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_a"/>
    <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_g"/>
    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_attr_d_value>
    <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_edge_type>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_ba -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_ba">
    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Edge"/>
    <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_b"/>
    <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_a"/>
    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_attr_d_value>
    <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_edge_type>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_bc -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_bc">

```

```

    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
    <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_b"/>
    <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_c"/>
    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_a
ttr_d_value>
    <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</ha
s_edge_type>
  </owl:NamedIndividual>

```

```

<!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_cb -->

```

```

  <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_cb">
    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
    <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_c"/>
    <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_b"/>
    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_a
ttr_d_value>
    <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</ha
s_edge_type>
  </owl:NamedIndividual>

```

```

<!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_ch -->

```

```

  <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_ch">
    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>

```

```

        <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_c"/>
        <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_h"/>
        <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</has_a
ttr_d_value>
        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</ha
s_edge_type>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_ea -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_ea">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
        <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_e"/>
        <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_a"/>
        <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_a
ttr_d_value>
        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</ha
s_edge_type>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_ef -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_ef">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
        <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_e"/>

```

```

    <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_f"/>
    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_attr_d_value>
    <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_edge_type>
  </owl:NamedIndividual>

```

```

  <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_fe -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_fe">
    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Edge"/>
    <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_f"/>
    <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_e"/>
    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_attr_d_value>
    <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_edge_type>
  </owl:NamedIndividual>

```

```

  <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_fg -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_fg">
    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Edge"/>
    <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_f"/>
    <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_g"/>

```

```

        <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_attr_d_value>
        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_edge_type>
    </owl:NamedIndividual>

```

```

<!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_fh -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_fh">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Edge"/>
        <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_f"/>
        <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_h"/>
        <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</has_attr_d_value>
        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_edge_type>
    </owl:NamedIndividual>

```

```

<!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_ga -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_ga">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Edge"/>
        <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_g"/>
        <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_a"/>
        <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_attr_d_value>

```

```

        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_
s_edge_type>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_gf -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_gf">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
        <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_g"/>
        <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_f"/>
        <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_a
ttr_d_value>
        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_
s_edge_type>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_gh -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_gh">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
        <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_g"/>
        <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_h"/>
        <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</has_a
ttr_d_value>
        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_
s_edge_type>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_hc -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_hc">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
            <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_h"/>
                <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_c"/>
                    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</has_a
ttr_d_value>
                        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</ha
s_edge_type>
                    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_hf -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#e_hf">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Edge"/>
            <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_h"/>
                <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#n_f"/>
                    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</has_a
ttr_d_value>
                        <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</ha
s_edge_type>
                    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#e_hg -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#e_hg">
    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Edge"/>
    <comes_from
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_h"/>
    <comes_to
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_g"/>
    <has_attr_d_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</has_attr_d_value>
    <has_edge_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">black</has_edge_type>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_a -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_a">
    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Node"/>
    <has_attr_n_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A</has_attr_n_value>
    <has_node_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">City</has_node_type>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_b -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_b">
    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Node"/>
    <has_attr_n_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">B</has_attr_n_value>

```

```

        <has_node_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">City</has
_node_type>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#n_c -->

        <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#n_c">
            <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Node"/>
                <has_attr_n_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">C</has_at
tr_n_value>
                    <has_node_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">City</has
_node_type>
            </owl:NamedIndividual>

            <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#n_e -->

                <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#n_e">
                    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Node"/>
                        <has_attr_n_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">E</has_at
tr_n_value>
                            <has_node_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">City</has
_node_type>
                    </owl:NamedIndividual>

                    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-
ontology-15#n_f -->

                        <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10
/untitled-ontology-15#n_f">
                            <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Node"/>

```

```

        <has_attr_n_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">F</has_attr_n_value>
        <has_node_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">City</has_node_type>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_g -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_g">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Node"/>
        <has_attr_n_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">G</has_attr_n_value>
        <has_node_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">City</has_node_type>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_h -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_h">
        <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#Node"/>
        <has_attr_n_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">H</has_attr_n_value>
        <has_node_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">City</has_node_type>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_st -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#n_st">

```

```

    <rdf:type
rdf:resource="http://www.semanticweb.org/student/ontologies/2023
/10/untitled-ontology-15#Node"/>
    <has_attr_max_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_a
ttr_max_value>
    <has_node_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">State</ha
s_node_type>
    </owl:NamedIndividual>
</rdf:RDF>

```

ПРИЛОЖЕНИЕ В

(Обязательное)

Реализация распределенной графо-трансформационной

системы на основе онтологий и языка SPARQL

SPARQL-запросы задачи поиска кратчайших путей

```
#PREFIX
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX :
<http://www.semanticweb.org/student/ontologies/2023/10/untitled-ontology-15#>
```

```
#QUERY ExpandStart
INSERT {
  [] rdf:type :Edge;
  :has_edge_type "blue"^^xsd:string;
  :comes_from ?c1;
  :comes_to ?c2;
  :has_attr_d_value ?x}
WHERE {
SELECT ?c1 ?c2 ?a3 ?a4 ?x
WHERE {
?c1 rdf:type :Node.
?c1 :has_node_type "City"^^xsd:string.
?c2 rdf:type :Node.
?c2 :has_node_type "City"^^xsd:string.
?a3 rdf:type :Edge.
?a3 :has_edge_type "start"^^xsd:string.
?a3 :comes_from ?c1.
?a3 :comes_to ?c1.
?a4 rdf:type :Edge.
?a4 :has_edge_type "black"^^xsd:string.
?a4 :comes_from ?c1.
?a4 :comes_to ?c2.
?a4 :has_attr_d_value ?x.
FILTER (?c1 != ?c2)
FILTER NOT EXISTS {
?a rdf:type :Edge.
?a :has_edge_type "blue"^^xsd:string.
?a :comes_from ?c1.
?a :comes_to ?c2}
}
LIMIT 1
}
```

```
#QUERY Expand
INSERT {
  [] rdf:type :Edge;
  :has_edge_type "blue"^^xsd:string;
  :comes_from ?c1;
  :comes_to ?c3;
  :has_attr_d_value ?yx}
WHERE {
SELECT ?c1 ?c3 ?yx
WHERE {
?c1 rdf:type :Node.
?c1 :has_node_type "City"^^xsd:string.
?c2 rdf:type :Node.
```

```

?c2 :has_node_type "City"^^xsd:string.
?c3 rdf:type :Node.
?c3 :has_node_type "City"^^xsd:string.
?s4 rdf:type :Node.
?s4 :has_node_type "State"^^xsd:string.
?s4 :has_attr_max_value ?m.
?a5 rdf:type :Edge.
?a5 :has_edge_type "black"^^xsd:string.
?a5 :comes_from ?c1.
?a5 :comes_to ?c3.
?a5:has_attr_d_value ?x.
?a6 rdf:type :Edge.
?a6 :has_edge_type "blue"^^xsd:string.
?a6 :comes_from ?c2.
?a6 :comes_to ?c1.
?a6:has_attr_d_value ?y.
BIND ((?y+?x) AS ?yx)
FILTER (?c1 != ?c2)
FILTER (?c1 != ?c3)
FILTER (?c2 != ?c3)
FILTER (?y+?x<?m)
# НАС-условия
FILTER NOT EXISTS {
?a rdf:type :Edge.
?a :has_edge_type "start"^^xsd:string.
?a :comes_from ?c1.
?a :comes_to ?c1}
FILTER NOT EXISTS {
?a rdf:type :Edge.
?a :has_edge_type "goal"^^xsd:string.
?a :comes_from ?c1.
?a :comes_to ?c1}
FILTER NOT EXISTS {
?a rdf:type :Edge.
?a :has_edge_type "start"^^xsd:string.
?a :comes_from ?c3.
?a :comes_to ?c3}
FILTER NOT EXISTS {
?a rdf:type :Edge.
?a :has_edge_type "blue"^^xsd:string.
?a :comes_from ?c1.
?a :comes_to ?c3}
} LIMIT 1
}

#QUERY expandMore
DELETE {
?a6 :has_attr_d_value ?z}
INSERT {
?a6 :has_attr_d_value ?yx}
WHERE {
SELECT ?a6 ?z ?yx
WHERE {

```

```

?c1 rdf:type :Node.
?c1 :has_node_type "City"^^xsd:string.
?c2 rdf:type :Node.
?c2 :has_node_type "City"^^xsd:string.
?c3 rdf:type :Node.
?c3 :has_node_type "City"^^xsd:string.
?a4 rdf:type :Edge.
?a4 :has_edge_type "black"^^xsd:string.
?a4 :comes_from ?c1.
?a4 :comes_to ?c3.
?a4:has_attr_d_value ?x.
?a5 rdf:type :Edge.
?a5 :has_edge_type "blue"^^xsd:string.
?a5 :comes_from ?c2.
?a5 :comes_to ?c1.
?a5:has_attr_d_value ?y.
?a6 rdf:type :Edge.
?a6 :has_edge_type "blue"^^xsd:string.
?a6 :comes_from ?c1.
?a6 :comes_to ?c3.
?a6 :has_attr_d_value ?z.
BIND ((?y+?x) AS ?yx)
FILTER (?c1 != ?c2)
FILTER (?c1 != ?c3)
FILTER (?c2 != ?c3)
FILTER (?y+?x<?z)
FILTER NOT EXISTS {
?a rdf:type :Edge.
?a :has_edge_type "start"^^xsd:string.
?a :comes_from ?c1.
?a :comes_to ?c1}
FILTER NOT EXISTS {
?a rdf:type :Edge.
?a :has_edge_type "goal"^^xsd:string.
?a :comes_from ?c1.
?a :comes_to ?c1}
FILTER NOT EXISTS {
?a rdf:type :Edge.
?a :has_edge_type "start"^^xsd:string.
?a :comes_from ?c3.
?a :comes_to ?c3}
} LIMIT 1
}

#QUERY setMax
DELETE {
?s3 :has_attr_max_value ?m}
INSERT {
?s3 :has_attr_max_value ?y}
WHERE {
SELECT ?s3 ?m ?y
WHERE {
?c1 rdf:type :Node.

```

```

?c1 :has_node_type "City"^^xsd:string.
?c2 rdf:type :Node.
?c2 :has_node_type "City"^^xsd:string.

?s3 rdf:type :Node.
?s3 :has_node_type "State"^^xsd:string.
?s3 :has_attr_max_value ?m.

?a4 rdf:type :Edge.
?a4 :has_edge_type "blue"^^xsd:string.
?a4 :comes_from ?c1.
?a4 :comes_to ?c2.
?a4:has_attr_d_value ?y.

?a5 rdf:type :Edge.
?a5 :has_edge_type "goal"^^xsd:string.
?a5 :comes_from ?c2.
?a5 :comes_to ?c2.
FILTER (?c1 != ?c2)
FILTER (?y<?m)
} LIMIT 1
}
#QUERY showRoad
DELETE {
?a rdf:type :Edge.
?a :has_edge_type "blue"^^xsd:string.
?a :comes_from ?c1.
?a :comes_to ?c2.
?a :has_attr_d_value ?z}
INSERT {
  [] rdf:type :Edge;
  :has_edge_type "red"^^xsd:string;
  :comes_from ?c1;
  :comes_to ?c2;
  :has_attr_d_value ?z}
WHERE {
SELECT ?c1 ?c2 ?a ?z
WHERE {
?c1 rdf:type :Node.
?c1 :has_node_type "City"^^xsd:string.
?c2 rdf:type :Node.
?c2 :has_node_type "City"^^xsd:string.
?s3 rdf:type :Node.
?s3 :has_node_type "State"^^xsd:string.
?s3 :has_attr_max_value ?m.
?a5 rdf:type :Edge.
?a5 :has_edge_type "goal"^^xsd:string.
?a5 :comes_from ?c2.
?a5 :comes_to ?c2.
?a rdf:type :Edge.
?a :has_edge_type "blue"^^xsd:string.
?a :comes_from ?c1.
?a :comes_to ?c2.

```

```

?a:has_attr_d_value ?z.
FILTER (?z = ?m)
} LIMIT 1
}

#QUERY showRoad1
DELETE {
  ?ax rdf:type :Edge.
  ?ax :has_edge_type "blue"^^xsd:string.
  ?ax :comes_from ?c3.
  ?ax :comes_to ?c1.
  ?ax :has_attr_d_value ?y}
INSERT {
  [] rdf:type :Edge;
  :has_edge_type "red"^^xsd:string;
  :comes_from ?c3;
  :comes_to ?c1;
  :has_attr_d_value ?y}
WHERE {
  SELECT ?ax ?c1 ?c3 ?y
  WHERE {
    ?c1 rdf:type :Node.
    ?c1 :has_node_type "City"^^xsd:string.
    ?c2 rdf:type :Node.
    ?c2 :has_node_type "City"^^xsd:string.
    ?c3 rdf:type :Node.
    ?c3 :has_node_type "City"^^xsd:string.
    ?a4 rdf:type :Edge.
    ?a4 :has_edge_type "black"^^xsd:string.
    ?a4 :comes_from ?c1.
    ?a4 :comes_to ?c2.
    ?a4:has_attr_d_value ?x.
    ?a5 rdf:type :Edge.
    ?a5 :has_edge_type "red"^^xsd:string.
    ?a5 :comes_from ?c1.
    ?a5 :comes_to ?c2.
    ?a5:has_attr_d_value ?z.
    ?ax rdf:type :Edge.
    ?ax :has_edge_type "blue"^^xsd:string.
    ?ax :comes_from ?c3.
    ?ax :comes_to ?c1.
    ?ax :has_attr_d_value ?y.
    FILTER (?c1 != ?c2)
    FILTER (?c1 != ?c3)
    FILTER (?c2 != ?c3)
    FILTER ((?z-?x)=?y)
    FILTER NOT EXISTS {
      ?a rdf:type :Edge.
      ?a :has_edge_type "start"^^xsd:string.
      ?a :comes_from ?c1.
      ?a :comes_to ?c1}
    FILTER NOT EXISTS {
      ?a rdf:type :Edge.

```

```

?a :has_edge_type "goal"^^xsd:string.
?a :comes_from ?c1.
?a :comes_to ?c1}
FILTER NOT EXISTS {
?a rdf:type :Edge.
?a :has_edge_type "goal"^^xsd:string.
?a :comes_from ?c3.
?a :comes_to ?c3}
} LIMIT 1
}

#QUERY removeExpand
DELETE {
?ax rdf:type :Edge.
?ax :has_edge_type "blue"^^xsd:string.
?ax :comes_from ?c1.
?ax :comes_to ?c2}
WHERE {
SELECT ?ax ?c1 ?c2
WHERE {
?c1 rdf:type :Node.
?c1 :has_node_type "City"^^xsd:string.
?c2 rdf:type :Node.
?c2 :has_node_type "City"^^xsd:string.
?ax rdf:type :Edge.
?ax :has_edge_type "blue"^^xsd:string.
?ax :comes_from ?c1.
?ax :comes_to ?c2
FILTER (?c1 != ?c2)
} LIMIT 1
}

#END

```

ПРИЛОЖЕНИЕ Г

(Обязательное)

Реализация распределенной графо-трансформационной

системы на основе онтологий и языка SPARQL

Онтология вычисления суммы целых чисел

```

<?xml version="1.0"?>
<rdf:RDF
xmlns="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-9#"

xml:base="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-9"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-9"/>

    <!--

////////////////////////////////////
////////////////////////////////////
    //
    // Data properties
    //

////////////////////////////////////
////////////////////////////////////
    -->

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-9#has_been_used -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-9#has_been_used"/>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-9#has_total_value -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-9#has_total_value"/>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-9#has_value -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-9#has_value"/>

```

```

<!--

////////////////////////////////////
////////////////////////////////////
//
// Classes
//

////////////////////////////////////
////////////////////////////////////
-->

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-9#Number -->

<owl:Class
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-9#Number"/>

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-9#SUM -->

<owl:Class
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-9#SUM"/>

<!--

////////////////////////////////////
////////////////////////////////////
//
// Individuals
//

////////////////////////////////////
////////////////////////////////////
-->

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-9#n1 -->

<owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-9#n1">
  <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-9#Number"/>

```

```

        <has_been_used
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</h
as_been_used>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_v
alue>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-9#n2 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-9#n2">
        <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-9#Number"/>
        <has_been_used
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</h
as_been_used>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_v
alue>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-9#n3 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-9#n3">
        <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-9#Number"/>
        <has_been_used
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</h
as_been_used>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_v
alue>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-9#n4 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-9#n4">

```

```

        <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-9#Number"/>
        <has_been_used
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</h
as_been_used>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</has_v
alue>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-9#n5 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-9#n5">
        <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-9#Number"/>
        <has_been_used
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</h
as_been_used>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">5</has_v
alue>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-9#s -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-9#s">
        <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-9#SUM"/>
        <has_total_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">0</has_t
otal_value>
    </owl:NamedIndividual>
</rdf:RDF>

```

ПРИЛОЖЕНИЕ Д

(Обязательное)

Реализация распределенной графо-трансформационной

системы на основе онтологий и языка SPARQL

Онтология умножения матрицы на вектор

```

<?xml version="1.0"?>
<rdf:RDF
xmlns="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11#"

xml:base="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    <owl:Ontology
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11"/>

    <!--

////////////////////////////////////
////////////////////////////////////
    //
    // Data properties
    //

////////////////////////////////////
////////////////////////////////////
    -->

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11#has_first_index -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11#has_first_index"/>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11#has_second_index -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11#has_second_index"/>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11#has_value -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-ontology-11#has_value"/>

```

```

<!--

////////////////////////////////////
////////////////////////////////////
//
// Classes
//

////////////////////////////////////
////////////////////////////////////
-->

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#A -->

<owl:Class
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#A"/>

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#B -->

<owl:Class
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#B"/>

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#C -->

<owl:Class
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#C"/>

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#Index -->

<owl:Class
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#Index"/>

<!--

////////////////////////////////////
////////////////////////////////////
//
// Individuals
//

```

```

////////////////////////////////////
////////////////////////////////////
-->

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#a11 -->

<owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#a11">
  <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#A"/>
    <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_f
irst_index>
      <has_second_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_s
econd_index>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_v
alue>
      </owl:NamedIndividual>

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#a12 -->

<owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#a12">
  <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#A"/>
    <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_f
irst_index>
      <has_second_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_s
econd_index>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</has_v
alue>
      </owl:NamedIndividual>

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#a13 -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#a13">
    <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#A"/>
    <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_f
irst_index>
    <has_second_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_s
econd_index>
    <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">0</has_v
alue>
    </owl:NamedIndividual>

```

```

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#a21 -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#a21">
    <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#A"/>
    <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_f
irst_index>
    <has_second_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_s
econd_index>
    <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">-
2</has_value>
    </owl:NamedIndividual>

```

```

<!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#a22 -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#a22">
    <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#A"/>
    <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_f
irst_index>

```

```

        <has_second_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_s
econd_index>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_v
alue>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#a23 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#a23">
        <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#A"/>
        <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_f
irst_index>
        <has_second_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_s
econd_index>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_v
alue>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#b1 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#b1">
        <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#B"/>
        <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_f
irst_index>
        <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_v
alue>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#b2 -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#b2">
    <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#B"/>
    <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_f
irst_index>
    <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_v
alue>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#b3 -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#b3">
    <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#B"/>
    <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">3</has_f
irst_index>
    <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">-
1</has_value>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#c1 -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#c1">
    <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#C"/>
    <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</has_f
irst_index>
    <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">999</has
_value>
    </owl:NamedIndividual>

```

```

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#c2 -->

```

```

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#c2">
    <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#C"/>
    <has_first_index
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</has_f
irst_index>
    <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">999</has
_value>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#i -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#i">
    <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#Index"/>
    <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">999</has
_value>
    </owl:NamedIndividual>

    <!--
http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#j -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/dubin/ontologies/2024/2/un
titled-ontology-11#j">
    <rdf:type
rdf:resource="http://www.semanticweb.org/dubin/ontologies/2024/2
/untitled-ontology-11#Index"/>
    <has_value
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">999</has
_value>
    </owl:NamedIndividual>
</rdf:RDF>

```

ПРИЛОЖЕНИЕ Е

(Обязательное)

Реализация распределенной графо-трансформационной

системы на основе онтологий и языка SPARQL

SPARQL-запросы умножения матрицы на вектор

```

#PREFIX
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX :
<http://www.semanticweb.org/dubin/ontologies/2024/2/untitled-
ontology-11#>

#QUERY SET_i_to_1    Установка индекса i в единицу
DELETE {:i :has_value ?zi}
INSERT {:i :has_value 1}
WHERE
{:i :has_value ?zi}

#QUERY SET_j_to_1    Установка индекса j в единицу
DELETE {:j :has_value ?zj}
INSERT {:j :has_value 1}
WHERE
{:j :has_value ?zj}

#QUERY SET_ci_to_0    Установка элемента вектора c(i) в ноль
DELETE {?c :has_value ?v}
INSERT {?c :has_value 0}
WHERE
{:i :has_value ?zi.
?c rdf:type :C.
?c :has_first_index ?zi.
?c :has_value ?v}

#QUERY INC_i    Инкремент индекса i
DELETE {:i :has_value ?zi}
INSERT {:i :has_value ?newzi}
WHERE
{:i :has_value ?zi.
BIND(?zi+1 AS ?newzi)}

#QUERY INC_j    Инкремент индекса j
DELETE {:j :has_value ?zj}
INSERT {:j :has_value ?newzj}
WHERE
{:j :has_value ?zj.
BIND(?zj+1 AS ?newzj)}

#QUERY CALC_ci    Вычисление c(i) по формуле c(i)=
c(i)+a(i,j)*b(j)
DELETE {?c :has_value ?vc}
INSERT {?c :has_value ?newvc}
WHERE
{:i :has_value ?zi.
:j :has_value ?zj.
?a rdf:type :A.
?b rdf:type :B.
?c rdf:type :C.
?a :has_first_index ?zi.

```

```

?a :has_second_index ?zj.
?b :has_first_index ?zj.
?c :has_first_index ?zi.
?a :has_value ?va.
?b :has_value ?vb.
?c :has_value ?vc.
BIND(?vc+?va*?vb AS ?newvc) }

```

```

#QUERY COMP_i      Сравнение индекса i с границей m=2 (на меньше
или равно)
ASK
WHERE
{:i :has_value ?zi.
FILTER(?zi<=2) }

```

```

#QUERY COMP_j      Сравнение индекса j с границей n=3 (на меньше
или равно)
ASK
WHERE
{:j :has_value ?zj.
FILTER(?zj<=3) }
#END

```

ПРИЛОЖЕНИЕ Ж

(Обязательное)

Реализация распределенной графо-трансформационной
системы на основе онтологий и языка SPARQL

Презентация

РЕАЛИЗАЦИЯ РАСПРЕДЕЛЕННОЙ ГРАФО- ТРАНСФОРМАЦИОННОЙ СИСТЕМЫ ТРАНСФОРМАЦИИ ГРАФОВ С ИСПОЛЬЗОВАНИЕМ ОНТОЛОГИЙ И ЯЗЫКА SPARQL

Автор работы: магистрант группы 22ВВим1 Ручкин М.А.

Научный руководитель: д.т.н. профессор Дубинин В.Н.

Пенза, 2024

Актуальность

Графы и их производные модели играют ключевую роль в информатике и вычислительной технике, обеспечивая описание и анализ структуры систем. Ограниченность статичных графов в динамических процессах подчеркивает необходимость разработки методов преобразования графов, что актуально в рамках MDE и семантического Веба. Семантический Веб и его принципы, такие как использование онтологий и структурированных данных, вдохновляют на создание новых методов трансформации графовых моделей для эффективной обработки и анализа данных. В последние годы наблюдается рост интереса к этим технологиям в разработке информационных обрабатывающих систем и систем автоматизированного проектирования различного назначения в среде семантического Веба, что подтверждает актуальность исследования по разработке распределенной графо-трансформационной системы на основе онтологий и языка SPARQL.

Цель и задачи

Цель работы - разработка методического, лингвистического, информационного и программного обеспечения для поддержки и реализации графо-трансформационного подхода к проектированию сложных систем в среде семантического Веб

Задачи исследования

- 1) Разработать онтологию типизированных атрибутивных графов
- 2) Разработать методику реализации правил трансформации графов на языке SPARQL
- 3) Разработать язык описания моделей выполнения SPARQL Update – правил
- 4) Разработать инструментальную программную систему для поддержки трансформации графов на основе RDF-онтологий и языка SPARQL
- 5) Реализовать конкретные системы трансформации графов с использованием разработанного инструментария

3

Научная новизна

1. Разработана онтология типизированных атрибутивных графов (ТАГ), в основу которой положено их математическое определение. Особенностью онтологии является словарь, пригодный для описания любых ТАГ, что позволяет использовать онтологию при разработке автоматизированных систем, включающих обработку графов, на основе технологий семантического Веб
2. Разработана методика кодирования правил трансформации графов в виде SPARQL Update - запросов. Особенностью методики является использование паттерна DELETE-INSERT-SELECT-LIMIT, что позволяет одним запросом описывать одновременно и правую, и левую часть правила преобразования графа;
3. Предложены модели выполнения SPARQL Update – правил. Особенностью которых является возможность реализации задач сложной обработки данных, характерной для языков программирования высокого уровня;
4. Разработан язык описания пользовательских моделей выполнения SPARQL Update – правил, позволяющий разрабатывать произвольные процедуры обработки RDF – данных, что значительно повышает возможности языка SPARQL и позволяет построение информационных обрабатывающих систем в среде семантического Веб.

4

Практические результаты

1. Разработана инструментальная программная система для поддержки трансформации графов на основе RDF-онтологий и языка SPARQL, позволяющая пользователям более гибко настраивать выполнение запросов и адаптировать их под конкретные задачи;
2. Разработаны онтология и правила преобразования для решения задачи поиска кратчайшего пути в графе, позволяющие оценить полезность использования онтологий и языка SPARQL в системах трансформации графов;
3. Показана применимость разработанной инструментальной программной системы для проведения числовых вычислений с использованием встроенного в эту систему языка описания моделей выполнения SPARQL Update – правил и собственно языка SPARQL.

В качестве тестовых примеров рассматривались:

- 1) вычисление суммы целых чисел,
- 2) умножение матрицы на вектор.

5

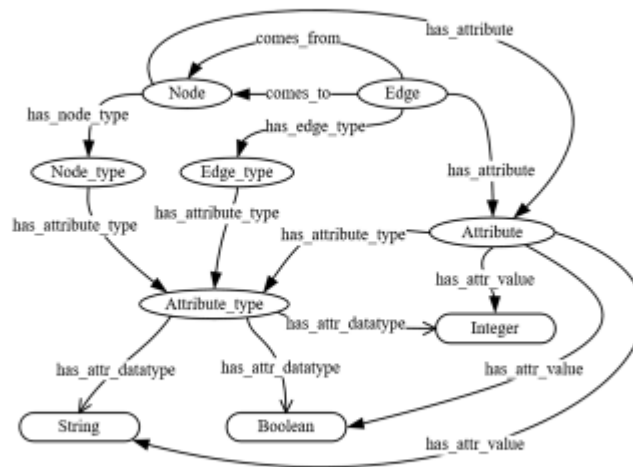
Онтология типизированных атрибутных графов

$$G = (N, E, NT, ET, AT, A, D, tn, te, tan, tae, at, na, ea, dom, val),$$

где N – множество вершин (узлов) графа,
 $E \subseteq N \times N$ – множество направленных ребер (дуг) графа,
 NT – множество типов узлов,
 ET – множество типов дуг,
 AT – множество типов (иначе, имен) атрибутов,
 A – множество (экземпляров) атрибутов,
 D – множество доменов (областей значений данных),
 $tn: N \rightarrow NT$ – сюръективное отображение, назначающее узлам соответствующие типы,
 $te: N \rightarrow ET$ – сюръективное отображение, назначающее дугам соответствующие типы,
 $tan: NT \rightarrow 2^{AT}$ – функция, назначающая типам узлов множества типов атрибутов,
 $tae: ET \rightarrow 2^{AT}$ – функция, назначающая типам дуг множества типов атрибутов,
 $at: A \rightarrow AT$ – сюръективное отображение, назначающее атрибутам соответствующие типы.
 $na: N \rightarrow 2^A$ – функция, назначающая узлам множества атрибутов,
 $ea: E \rightarrow 2^A$ – функция, назначающая дугам множества атрибутов,
 $dom: AT \rightarrow D$ – функция, определяющая область значений типов атрибутов,
 $val: A \rightarrow D_1 \cup D_2 \cup \dots \cup D_r$ – функция значений атрибутов.

6

ТВох-онтология ТАГ



7

Соответствие математических объектов элементам онтологии

Математический объект	Элемент онтологии
Множество N	Класс Node
Множество E	Класс Edge
Множество NT	Класс Node_type
Множество ET	Класс Edge_type
Множество A	Класс Attribute
Множество AT	Класс Attribute_type
Сюръективное отображение tn	Объектное свойство has_node_type
Сюръективное отображение te	Объектное свойство has_edge_type
Функции tap и tae	Объектное свойство has_attribute_type
Сюръективное отображение at	Объектное свойство has_attribute_type
Функции na и ea	Объектное свойство has_attribute
Функция dom	Свойство по данным has_attr_datatype
Функция val	Свойство по данным has_attr_value

8

Методика кодирования правил трансформации графов в виде SPARQL Update - запросов

Левая часть ПТГ вместе с NAC- и AC-условиями представляется в WHERE-предложении с вложенным подзапросом, а именно в виде следующей конструкции:

```
WHERE {  
  SELECT <список параметров>  
  WHERE {<граф левой части ПТГ>, <NAC-условия>, <AC-  
    условия>}  
  LIMIT 1}
```

При этом <список параметров> есть список переменных, представляющих вершины и дуги графа (а также их атрибуты) из левой части ПТГ, участвующих в правой части ПТГ и, соответственно в DELETE- и INSERT-предложениях SPARQL Update-запросов.

Правая часть ПТГ представляется с помощью DELETE- и INSERT-предложений SPARQL Update-запросов. Предложение DELETE используется для удаления вершин и дуг графа, а предложение INSERT – для их создания. В совокупности конструкция DELETE- INSERT используется для модификации значений атрибутов вершин и дуг графа.

9

Модели выполнения SPARQL Update -правил

Для автоматического управления выполнением SPARQL Update-правил могут использоваться два класса моделей:

- 1) предопределенные элементарные модели выполнения,
- 2) модели выполнения, определяемые пользователем.

Для реализации моделей выполнения используются два основных типа запросов:

- 1) SPARQL Update-запрос, имеющий структуру DELETE-INSERT-WHERE. По сути дела, это специализированное продукционное правило ЕСЛИ <Условие>, ТО <Действие>.,
- 2) ASK-запрос, содержащий в своей структуре только <Условие> (предложение WHERE).

10

Ациклическая однопроходная модель выполнения (OAS)

Правила выполняются последовательно и однократно, одно за другим, за один проход.

Спецификация: OAS(R1,R2,...,Rn).

Интерпретация на псевдокоде:

```
    ехес R1;  
    ехес R2;  
    ...  
    ехес Rn.
```

11

Ациклическая многопроходная модель выполнения (OAM)

Правила выполняются последовательно и однократно, одно за другим, за несколько проходов, до тех пор, пока все правила не окажутся не разрешенными.

Спецификация: OAM(R1,R2,...,Rn).

Интерпретация на псевдокоде:

```
repeat  
    ехес R1;  
    ехес R2;  
    ...  
    ехес Rn  
until not(A(R1) v A(R2) v ... v A(Rn)).
```

12

Циклическая однопроходная модель выполнения (OCS)

Правила выполняются последовательно многократно, одно за другим, за один проход. Каждое правило выполняется до тех пор, пока оно является разрешенным.

Спецификация: $OCS(R1, R2, \dots, Rn)$.

Интерпретация на псевдокоде:

```
while A(R1) exec R1;  
while A(R1) exec R2;  
...  
while A(R1) exec Rn.
```

13

Циклическая многопроходная модель выполнения (OCM)

Правила в наборе выполняются последовательно циклически, одно за другим, за несколько проходов, до тех пор, пока все правила не окажутся не разрешенными. В рамках данной модели в отдельном проходе каждое правило выполняется до тех пор, пока оно является разрешенным.

Спецификация: $OCM(R1, R2, \dots, Rn)$.

Интерпретация на псевдокоде:

```
repeat  
    while A(R1) exec R1;  
    while A(R1) exec R2;  
    ...  
    while A(R1) exec Rn;  
until not(A(R1)∨A(R2)∨...∨A(Rn)).
```

14

Приоритетная модель выполнения (PRIOR)

Правила в наборе выполняются в соответствии с приоритетом, который определяется положением правила в наборе (то есть, линейным порядком) – чем ближе к началу списка, тем приоритетнее правило.

Спецификация: PRIOR(R1,R2,...,Rn).

Интерпретация на псевдокоде:

```
repeat
  repeat
    exec R1
  until not(A(R1))
  exec R2
until not(A(R1)∨A(R2))
...
exec Rn
until not(A(R1)∨A(R2)∨...∨A(Rn)).
```

15

Рандомная модель выполнения (RAND)

Правило для выполнения выбирается случайным образом. Выполнение правил заканчивается только тогда, когда не остается ни одного разрешенного правила.

Спецификация: RAND(R1,R2,...,Rn).

Интерпретация на псевдокоде:

```
repeat
  c=rand(1,n)
  case c
    1: exec R1;
    2: exec R2;
    ...
    n: exec Rn;
until not(A(R1)∧A(R2)∧...∧A(Rn)).
```

16

Пользовательские модели выполнения на языке SPARQL/C

Пользователь может сформировать свою модель выполнения правил с использованием собственного простого языка SPARQL/C интерпретирующего типа, включающего следующие операторы:

- составной оператор do-end,
- условный оператор if-then-else,
- оператор цикла repeat-until,
- оператор цикла while.

Особенностью данных операторов в контексте моделей выполнения правил является использование в них трех действий (частично описанных выше):

- exec Ri – запуск i-ro SPARQL-UPDATE-запроса,
- A(Ri) – определение того, есть ли решения для i-ro SPARQL-UPDATE-запроса (реализуется ASK-запросом),
- C(Ri) – число найденных решений для i-ro SPARQL-UPDATE-запроса (реализуется COUNT-запросом).

17

Примеры на языке SPARQL/C

```
Пример 1:  
  if A(R1) then  
    exec R1  
  else  
    exec R2
```

```
Пример 3:  
  if A(R1) then  
    OAS(R1,R2)  
  else  
    OAS(R3,R4)
```

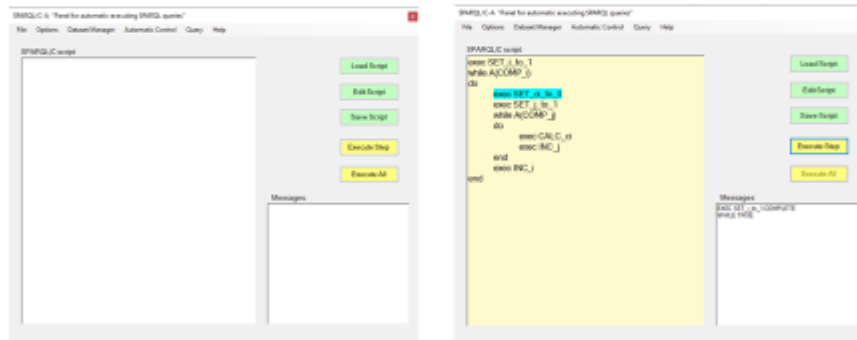
```
Пример 2:  
  if A(R1) then  
    do  
      exec R1  
      exec R2  
    end  
  else  
    do  
      exec R3  
      exec R4  
    end
```

```
Пример 4:  
  if A(R1) then  
    OAS(R1,R2)  
  else  
    OAS(R3,R4)  
  while A(RQX1) or not A(DD9) and A(R3)  
  do  
    exec rule_Load  
    exec rule_Add  
    exec rule_Mult  
    exec rule_Store  
  end
```

18

Инструментальная программная системы для поддержки трансформации графов SPARQL/C-A

SPARQL/C-A – это программная система для автоматического выполнения SPARQL UPDATE – запросов. Запуском правил управляет специальная программа (скрипт), написанная на языке SPARQL/C.



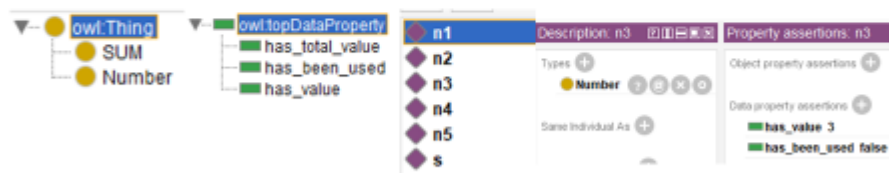
19

Онтология вычисления суммы целых чисел

Онтология состоит из двух классов, где Number – это число, а SUM – это сумма. Можно определить следующие свойства по данным:

- has_value – иметь значение для числа (целое),
- has_been_used – признак использования значения числа при вычислении суммы (false – число не использовалось, true – число было использовано),
- has_total_value – иметь значение для суммы (целое).

В онтологию были добавлены индивиды, $n1..n5$ соответственно числа со значениями от одного до пяти, а s – это сумма.



20

SPARQL-запросы вычисления суммы целых чисел

Запрос *ADD* – это основной SPARQL Update-запрос в данном примере. Этот запрос добавляет одно конкретное значение числа к сумме. После добавления к сумме число помечается как «использованное».

```
DELETE {  
  :s :has_total_value ?sum.  
  ?n :has_been_used false}  
INSERT {  
  :s :has_total_value ?newsum.  
  ?n :has_been_used true}  
WHERE {SELECT ?n ?sum ?newsum  
WHERE  
  {?n :has_value ?v.  
   ?n :has_been_used false.  
   :s :has_total_value ?sum.  
   BIND (?sum+?v AS ?newsum)}} LIMIT 1}
```

Запрос *TEST* – это ASK-запрос, определяющий есть ли неиспользованные числа для суммирования.

```
ASK  
{?n :has_been_used false}
```

21

Апробация работы

Основные научные положения и результаты работы докладывались и обсуждались на следующих научно-технических конференциях:

7-й Международной научно-технической конференции «Современные технологии в науке и образовании (СТНО)» (г. Рязань, 2024),

XI Ежегодной всероссийской межвузовской научно-практической конференции «Информационные технологии в науке и образовании. Проблемы и перспективы (ВМНПК – 2024)» (г. Пенза, 2024).

22

Публикации

Ручкин А.М., Дубинин А.В., Дубинин В.Н. Онтология типизированных атрибутивных графов // Сб. ст. XI Ежегодн. всеросс. межвуз. науч.-практ. конф. «Информационные технологии в науке и образовании. Проблемы и перспективы (ВМНПК – 2024)», Пенза, 2024.

Дубинин В.Н., Ручкин М.А., Дубинин А.В., Климкина Л.П. Использование языка SPARQL в реализации графодинамической модели для поиска кратчайших путей в графе // Современные технологии в науке и образовании – СТНО-2024 [текст]: сб. тр. VII междунар. науч.-техн. форума: в 10 т. Т.5./ под общ. ред. О.В. Миловзорова. – Рязань: Рязан. гос. радиотехн. ун-т, 2024. – С. 160-165.

23

Спасибо за внимание!

24