МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**Национальный исследовательский ядерный университет «МИФИ»**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Факультет кибернетики и информационной безопасности

Кафедра системного анализа

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К УЧЕБНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ**

**И КУРСОВОМУ ПРОЕКТУ ПО КОРПОРАТИВНЫМ ИНФОРМАЦИОННЫМ СИСТЕМАМ**

**НА ТЕМУ:**

|  |
| --- |
|  |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Студент* |  | | | | (Бондаренко Ю. А.) |
| *Руководитель* | | |  | | (Максимов Н. В.) |
| *Консультант* | | |  | |  |
| *Оценка* | |  | | | |
| *Подписи членов экзаменационной комиссии* | | | |  | |

Москва – 2016 г.

**Содержание**

Список используемых сокращений3

Введение4

1. Теоретическая часть работы5
   1. Определение онтологий5
   2. Операции над онтологиями8
      1. Операция объединения10
      2. Операция пересечения11
      3. Операция проекции11
      4. Операция масштабирования12
2. Практическая часть работы13
   1. Формирование базы данных13
      1. Выделение необходимых свойств графа13
      2. Способы задания данных13
      3. Структура базы данных15
   2. Выбор графической библиотеки16
   3. Реализация операций над онтологиями20
      1. Операция объединения 20
      2. Операция пересечения22
3. Заключение23
4. Список используемой литературы23
5. Приложение24

**Список используемых сокращений**

ПОЗ – поисковой образ запроса

ПОД – поисковой образ документа

ПрО – предметная область

ИПС – информационно-поисковая система

JS – язык программирования JavaScript

БД – база данных

**Введение**

Будем считать, что каждый из нас может иметь свои представления в некоторой области знания. И перед каждым из нас рано или поздно возникает задача предъявить эти представления как самому себе, так и другим людям. Одним из современных направлений решения этой задачи является графическое изображение структуры этих представлений с помощью компьютера.

Актуальность выбранной темы обусловлена необходимостью человека в обучении и саморазвитии, и, как следствие, в визуальном представлении знаний, которое значительно облегчает эту задачу.

Целью настоящей работы является визуализация графового представления онтологического подхода к документальной информации.

В соответствии с поставленной целью, необходимо решить следующие задачи:

– исследование теоретических основ работы с онтологическим представлением знаний;

– формирование оптимальной структуры базы данных, необходимых для визуализации графового представления онтологий;

– сравнительный анализ графических библиотек языка JS, с помощью которых возможно реализовать визуализацию графового представления онтологий, и выбор наиболее «подходящей» для данной задачи;

– разработка визуального представления онтологий и операций над ними.

1. **Теоретическая часть работы**
   1. **Определение онтологий**

Рассматривая онтологию ПрО с узкой точки зрения идентификации содержания документа при информационном поиске, будем представлять её как совокупность трех систем – функциональной, понятийной и терминологической.

Функциональная система («рабочий интерфейс» онтологии в деятельности субъекта) представляет объекты ПрО и отношения между ними средствами знакового уровня.

Эти отношения имеют функциональную окраску, т.к. определяют существенные с точки зрения задач пользователя способы и характер совместного существования и использования объектов.

Однако функциональная система не может рассматриваться вне связи со сложившейся в ПрО на *текущий момент времени* системой понятий, которая обычно фиксируется в форме тезаурусов, рубрикаторов, классификационных схем (логико-семантический базис онтологии).

В понятийной системе объектами (представленными также средствами знакового уровня) являются устойчивые понятия ПрО, а набор отношений ограничен родовидовыми и ассоциативными.

Терминологическая система в онтологии отражает свойства ЕЯ на уровне знаков -терминов ПрО, которые могут быть связаны отношениями эквивалентности (синонимии) и включения (образования словосочетаний). В качестве термина выступает отдельное слово или словосочетание ЕЯ (а также шифр классификации), которое может быть использовано для описания понятия или объекта в рамках заданной ПрО.

Для определения (установления) взаимосвязи этих систем (а также и их влияния друг на друга в процессе развития) может быть положена простая операция тождества элементов на знаковом уровне.

Таким образом, некоторая онтология ПрО (с точки зрения задач информационного поиска) формально может быть определена, как:

*O =<* , , , ≡ *>*, где

*–* функциональная система;

*–* понятийная система;

*–* терминологическая система;

≡ – операция сопоставления элементов различных систем на уровне знаков, обеспечивающая их тождество в функциональной, понятийной и терминологической системах.

Определенная таким образом онтология также обладает свойствами системы (с т.з. ОТС), т.е. для отдельной предметной области (в том числе и отдельного её элемента) может быть построено столько онтологий, сколько может быть аспектов рассмотрения ПрО, т.е. . Однако с целью упрощения выкладок, если не будет явной необходимости, мы будем опускать соответствующий индекс.

Отметим, что в качестве функционального (и терминологического) компонента может быть и система понятий ПрО. В этом случае её понятийной системой будет система метапонятий (это отвечает, например, определению «метаонтологии», где на верхнем уровне есть лишь три категории: «объект», «процесс» и «роль»).

Рассмотрим компоненты, входящие в приведенное выше определение онтологии.

*Функциональная система* определяется как:

*=<* , , , *>,* где

– множество знаковых описаний объектов ПрО;

– множество характеристических атрибутов;

– множество функциональных отношений;

– закон композиции, в соответствии с которым выбрано конкретное системное основание {, , }.

Со структурной точки зрения функциональная система может быть представлена помеченным взвешенным направленным мультиграфом

= <, >, где

- множество вершин, ;

- множество дуг, .

Каждая дуга задается тройкой:

, где

– вершина начала дуги;

– вершина завершения дуги;

– вес дуги (идентификатор функционального отношения).

Направленный мультиграф может иметь дуги и , , но не может при этом содержать дугу

*Понятийная система* определяется как:

*=<* , , , *>,* где

– множество знаковых описаний понятий ПрО;

– множество характеристических атрибутов знаковых описаний;

– родо-видовые и ассоциативные отношения;  
– закон композиции, в соответствии с которым выбрано конкретное системное основание {, , }.

Со структурной точки зрения понятийная система представлена помеченным взвешенным направленным графом и (с точки зрения введения операций над онтологиями) описана как

= <, >, где

- множество вершин, ;

- множество дуг, .

Каждая дуга задается тройкой:

, где

- вершина начала дуги;

- вершина завершения дуги;

- вес дуги (идентификатор отношения).

Направленный граф не содержит симметричных дуг, любые две вершины могут быть соединены только одной дугой.

*Терминологическая система*:  
*=<* , , , *>*  
– множество терминов ПрО;  
– множество характеристических атрибутов терминов;  
– отношения эквивалентности и включения;  
– закон композиции, в соответствии с которым выбрано конкретное системное основание {, , }.

Со структурной точки зрения терминологическая система описывается *n*-связным графом , где каждая компонента связности представляет собой полный граф (эквивалентность), дерево (включение) или результат операции объединения полных графов и деревьев (при наличии общих вершин).

* 1. **Операции над онтологиями**

Исследования в области онтологических представлений знаний интенсивно, хотя и достаточно эклектично развивающееся направление. Это в значительной степени относится и к введению операций над онтологиями. В многочисленных работах предлагаются достаточно разнообразные множества операций, которые достаточно условно можно отнести к классам:

1. Аналоги теоретико-множественных операций (пересечение, объединение, разность);

2. Операции извлечения и удаления фрагментов онтологий для использования при создании новых;

3. Операции проверки логической эквивалентности онтологий, перевода онтологий на другой формальный язык и т.п.;

4. Операции-функции – средства поддержки онтологий в инструментальных системах (создание, копирование, вывод и т.д.).  
Рассматриваемые далее операции предназначены для построения структурных комбинаций информационных объектов (документов, сообщений), представленных онтологиями, на которых субъект может построить новое знание (задачи вывода нового знания на имеющихся онтологиях не рассматриваются).

В качестве основных операций над онтологиями (на структурном уровне) рассмотрим операции из 1-го и 2-го классов - бинарные операции объединения и пересечения и унарные - построения аспектного представления и масштабирования онтологий, с помощью которых можно, в том числе, синтезировать новые онтологии, отражающие предметную область в аспекте, заданном пользователем.

В силу того, что операции над онтологиями рассматриваются в рамках одной ПрО, будем считать, что исходные онтологии имеют общие понятийную и терминологическую системы. Для формализации операций над онтологиями необходимо определить функцию подобия, вычисляющую меру соответствия при сравнении понятий и связей в исходных онтологиях с целью выделения семантически схожих элементов.

Введем функцию подобия *sim*(*a*,*b*), обладающую следующими очевидными свойствами:

1. *Sim*(*a*, *b*) ∈ [0;1] ;

2. *Sim*(*a*, *b*) = 1 → *a* = *b* (элементы *a*, *b* тождественны);

3. *Sim*(*a*, *b*) = 0 → *a* ≠ *b* (элементы *a*, *b* различны);

4. *Sim*(*a*, *a*) = 1 (свойство возвратности);

5. *Sim*(*a*, *b*) = *Sim*(*b*, *a*) (свойство симметричности).

Функция подобия рассчитывается для элементов функциональных систем пары исходных онтологий *O1* и *O2* (в дальнейшем описании индекс *i1* характеризует принадлежность элемента онтологии *O1*, а индекс *i2* - онтологии *O2*) по следующим правилам.

Для вершин мультиграфов:

1. *sim* = 1 , если ≡

2. *sim* = , если в графе общей понятийной системы онтологий существуют принадлежащие одному маршруту вершины и такие, что ≡   
и (*n* – длина маршрута, ( *j* = ) - вес отдельной дуги).

3. *sim*, если в графе общей терминологической системы онтологий  
существует полный подграф, содержащий вершины и такие, что ≡   
и (*p* – количество вершин подграфа)

4. *sim*, если в графе общей терминологической системы онтологий  
существует цепь между вершинами и такими, что ≡   
и (*q* – длина цепи)

5. *sim*, если не применимо ни одно из правил 1-4.  
Для дуг мультиграфов:  
*sim*

Результат бинарных операций над онтологиями , , и  
, , , приведенными к одному понятийному и терминологическому основанию, представляет собой онтологию , , . Таким образом, необходимо формализовать бинарные операции для функциональных систем.

Представление функциональных систем как мультиграфов позволяет свести операции над онтологиями к операциям над мультиграфами и

При выполнении операций над мультиграфами будем считать одинаковыми вершинами те, для которых функция подобия отлична от 0, а одинаковыми дугами – те, для которых функция подобия равна 1.

* + 1. **Операция объединения онтологий .**

Формально алгоритм объединения можно представить следующим образом:

1. Вычисляется множество вершин (для вершин разных мультиграфов, принадлежащих множеству , функция подобия равна 1).

2. Для каждой вершины из множества вершин формируется множество маршрутов к вершинам из множества в понятийном графе . Если для вершины множество маршрутов не пусто, для каждого из маршрутов вычисляется функция подобия. Две вершины, для которых функция подобия принимает максимальное значение, считаются далее тождественными (в мультиграфах каждая из вершин замещается соответствующим «понятийным маршрутом») и при новом пересечении множеств вершин мультиграфов формируют множество с учетом понятийной системы.

3. Рассматриваются множества вершин и . При наличии пары вершин (по одной из каждого множества), входящих в одну компоненту связности графа терминологической системы, происходит замена каждой из вершин соответствующей компонентой (или ее фрагментом, включающим обе вершины) и при новом пересечении множеств вершин мультиграфов формируют множество с учетом терминологической системы.

4. Множество вершин мультиграфа объединения онтологий формируется как результат теоретико-множественных операций:

5. Инцидентность дуг из множеств и сохраняется в мультиграфе объединения онтологий (с учетом слияния дуг, для которых функция подобия равна 1).

При объединении функциональных систем исходных онтологий по такому алгоритму могут возникнуть противоречия двух типов.

1. В результате операции объединения мультиграф онтологии может содержать вершины из множеств и , которые являются подграфами понятийного и терминологического графов, дуги и ребра которых описывают отношения понятийной и терминологической систем (что противоречит определению мультиграфа функциональной системы).  
Для разрешения этих противоречий необходима экспертная оценка и замена таких подграфов узлами (и, возможно, дугами), соответствующими функциональной системе.

2. Сохранение в мультиграфе объединения онтологий инцидентности дуг из множеств и может привести к наличию противоречивых дуг, инцидентных одной и той же паре вершин. Такая ситуация также исследуется субъектом – экспертом.

* + 1. **Операция пересечения онтологий .**

Отличие алгоритма пересечения от алгоритма объединения состоит в способе формирования результирующего мультиграфа:

,

а инцидентность сохраняется в мультиграфе пересечения онтологий только для тех дуг из множеств и , для которых функция подобия равна 1.

При пересечении функциональных систем исходных онтологий могут возникнуть противоречия только первого из описанных выше типов.

* + 1. **Операция проекции онтологий (построение аспектного представления).**

Аспект рассмотрения (представления, описания) в свою очередь может быть задан функциональной системой *=<* , , , *>,*  тогда операция проекции может быть сведена к операции пересечения исходной – *O =<* , , , ≡ *>* и аспектной – , , онтологий:

.

В результате операции пересечения онтологий на самом деле происходит «обогащение» аспектного описания с учетом понятийной и терминологической систем.

Рассмотрим следующие возможные ситуации.

1**.**  - аспект задается на уровне объектов – знаковыми описаниями совокупности объектов. Мультиграф аспектной онтологии, участвующий в операции пересечения, при этом представляет собой пустой граф . В этом случае для множества вершин результирующего мультиграфа операция пересечения выполняется по правилам, описанным выше, а множество дуг формируется из дуг множества , инцидентных вершинам множества .

2. - аспект задается на функциональном уровне – множеством  
функциональных отношений. В этом случае множество дуг мультиграфа проекции , а множество вершин формируется из вершин, инцидентных дугам из множества .

3. - аспект задается на смешанном, объектно-функциональном уровне, и для операции пересечения может быть сформирован мультиграф с непустыми множествами вершин и дуг.

* + 1. **Операция масштабирования онтологий.**

Для описания операций масштабирования (укрупнения или детализации) онтологии определим для исходной онтологии *O =<* , , , ≡ *>*  
«онтологию масштабирования» как , , , где и сведем операции масштабирования к разрешению противоречий первого типа в онтологии в соответствии со следующими правилами:

- в случае укрупнения исходной онтологии: в узлах, содержащих дерево понятий,  
остаются понятия самого верхнего уровня, а в узлах, содержащих подграфы терминологической системы – отдельные слова;

- в случае детализации исходной онтологии: в узлах, содержащих дерево понятий,  
остаются понятия самого нижнего уровня, а в узлах, содержащих подграфы терминологической системы – словосочетания.

1. **Практическая часть работы**
   1. **Формирование базы данных**
      1. **Определение необходимых свойств графа**

В теоретической части работы были выделены свойства графовых представлений онтологий для различных систем.

Объединив все необходимые свойства, получаем:

Узлы должны обладать следующими свойствами:

– Вес

– Имя сущности

Дуги должны обладать следующими свойствами:

– Вес дуги (идентификатор отношения)

– Ориентированность

* + 1. **Способы задания данных для представления графа**

Существует несколько видов задания данных для представления графа, такие как:

*1. Матрица смежности*

Таблица, где как столбцы, так и строки соответствуют вершинам графа. В каждой ячейке этой матрицы записывается число, определяющее наличие связи от вершины-строки к вершине-столбцу (либо наоборот).

Это наиболее удобный способ представления плотных графов.

Недостатком являются требования к памяти, прямо пропорциональные квадрату количества вершин.

### *2. Матрица инцидентности*

Таблица, где строки соответствуют вершинам графа, а столбцы соответствуют связям (рёбрам) графа. В ячейку матрицы на пересечении строки i со столбцом j записывается:

*1* в случае, если связь j «выходит» из вершины i,

*−1*, если связь «входит» в вершину,

*0* во всех остальных случаях (то есть если связь является петлёй или связь не инцидентна вершине)

Данный способ является самым ёмким (количество занимаемой памяти пропорционально произведению количества вершин графа на количество его ребер) для хранения, поэтому применяется очень редко, в особых случаях (например, для быстрого нахождения циклов в графе).

### *3. Список смежности*

Список, где каждой вершине графа соответствует строка, в которой хранится список смежных вершин. Такая структура данных не является таблицей в обычном понимании, а представляет собой «список списков».

Размер занимаемой памяти пропорционален сумме количества вершин и количества ребер.

Это наиболее удобный способ для представления разреженных графов, а также при реализации базовых алгоритмов обхода графа в ширину или глубину, где нужно быстро получать «соседей» текущей просматриваемой вершины.

### *4. Список рёбер*

Список, где каждому ребру графа соответствует строка, в которой хранятся две вершины, инцидентные ребру.

Размер занимаемой памяти пропорционален количеству ребер графа.

Это наиболее компактный способ представления графов, поэтому часто применяется для внешнего хранения или обмена данными.

Однако, ни один из этих четырех способов не подходит для визуализации графовых представлений онтологий, так как:

а). размер потребляемых ресурсов представлением с помощью матриц слишком быстро растет при увеличении количества вершин и ребер.

б). для реализации операций над онтологиями необходимо определить такие свойства графа, как вес вершины, наименование вершины, вес ребра, наименование ребра, что не представляется возможным в рассматриваемых представлениях данных.

* + 1. **Структура базы данных**

Таким образом, для определения необходимых свойств было решено сформировать следующую структуру базы данных, которые необходимо визуализировать:

БД представляет собой файл формата json, в котором определены графы. Каждый граф определен как набор массивов его ребер и узлов.

Структура каждого узла графа представляет собой набор параметров:

– идентификатор;

– имя сущности, визуализируемой узлом графа;

– первоначальные координаты (x, y) узла графа на экране;

– вес узла.

Структура каждого ребра графа представляет собой набор параметров:

– идентификатор;

– текст на ребре графа;

– идентификатор вершины, из которого исходит ребро;

– идентификатор вершины, в которое входит это ребро.

Пример определения одного из графов в программной системе приведен в Приложении №1.

* 1. **Выбор графической библиотеки**

Перед тем, как начать реализовывать визуализация онтологического подхода к идентификации информации с помощью мультиграфов, необходимо было провести сравнительный анализ графических библиотек, которые поддерживает JS.

Было выделено несколько основных свойств графической библиотеки:

1. Возможность реализации визуализации мультиграфа;
2. Простота использования библиотеки;
3. Быстродействие;
4. Функциональные возможности;
5. Качество документации.

Рассматривались следующие графические библиотеки:

1. D3.js;
2. Arbor.js;
3. CytoScape.js.

Установка всех трех библиотек проходила идентичным образом: либо указанием прямой ссылки на файл в интернете *(<script src=https://d3js.org/d3.v3.min.js ></script>*), либо скачиванием необходимых файлов с официального сайта и указанием ссылки на него (*<script src="js/arbor.js"></script>* ).

Графическая библиотека D3.js призвана для обработки и визуализации данных. Функций в D3.js более чем достаточно: работа с 3D графикой, работа с графами, построение таблиц, различных динамических диаграмм, деревьев и даже карт.

Arbor.js является библиотекой для работы с графами с широким набором функций для работы с ними. Однако, если рассмотреть этот набор функций относительно их применимости к данной задаче, то выяснится, что он недостаточен для полноценной работы с графами

Cytoscape.js также является библиотекой для работы с графами, однако эта библиотека слишком сложна в освоении и имеет функционал, недостаточный для текущей задачи.

При сравнении библиотек необходимо ввести понятие веса критерия сравнения. Под весом критерия будем подразумевать его значимость (важность) для реализации текущей задачи. Таким образом, самыми значимыми будут критерии «Быстродействие» и «Функциональные возможности».

Рассмотрим эти два критерия.

1. Функциональные возможности

|  |  |  |  |
| --- | --- | --- | --- |
| Возможности графической библиотеки | D3.js | Arbor.js | CytoScape.js |
| Визуализация графов | + | + | + |
| Добавление надписей к вершинам графа | + | + | + |
| Добавление надписей к ребрам графа | + | - | - |
| Разнообразие стилей отрисовки вершин и ребер графа | + | + | - |
| Возможность масштабирования изображения графа | + | + | + |
| Возможность передвигать вершины | + | + | + |

Наибольшая оценка по критерию «Функциональные возможности» была поставлена библиотеке D3.js, так как по всем рассматриваемым признакам она либо превосходит другие библиотеки, либо эквивалентна им.

1. Быстродействие

Был проведен сравнительный анализ скорости отрисовки графов для всех трех библиотек на малопроизводительном компьютере. Ниже приведена диаграмма, показывающая, какую часть времени от 2,5 секунд потратила программа, использующая каждую из библиотек, на процесс визуализации графов (по оси ординат откладывается время в миллисекундах).

Как видно из приведенной выше диаграммы, меньше всего времени на отрисовку требует библиотека D3.js. Высокая скорость работы в D3 достигается модульностью своих частей, т.е. при желании нарисовать простой кружочек, не будут подгружаться также функции для рисовки 3D моделей и кривых Безье. Второе место укрепилось за Arbor.js. Самой же малопроизводительной оказалась библиотека Cytoscape.js. Так как визуально все три библиотеки работают без нареканий, были поставлены соответственно 10, 9 и 8 баллов по этому критерию.

Ниже приведена сравнительная таблица характеристик графических библиотек. Все оценки наименее весомых критериев (все, кроме основных: «функциональные возможности» и «быстродействие») были поставлены на основании субъективного мнения разработчика.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии | Вес критерия | Название оцениваемой библиотеки | | |
| D3.js | Arbor.js | CytoScape.js |
| Оценка | | |
| Простота использования библиотеки | 6 | 7 | 9 | 5 |
| Быстродействие | 10 | 10 | 9 | 8 |
| Функциональные возможности | 10 | 10 | 6 | 5 |
| Качество документации | 8 | 8 | 7 | 5 |
| Итого | | 306 | 260 | 200 |

Исходя из результатов оценивания графических библиотек, была выбрана библиотека D3.js

* 1. **Реализация операций над онтологиями**
     1. **Операция объединения**

Схема работы функции, выполняющей объединение двух онтологий graph1 и graph2

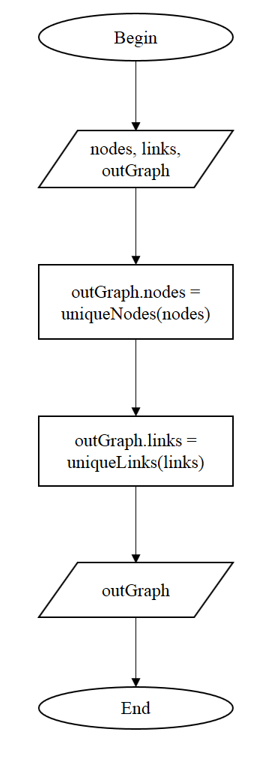


Схема работы функции uniqueNodes(nodes)

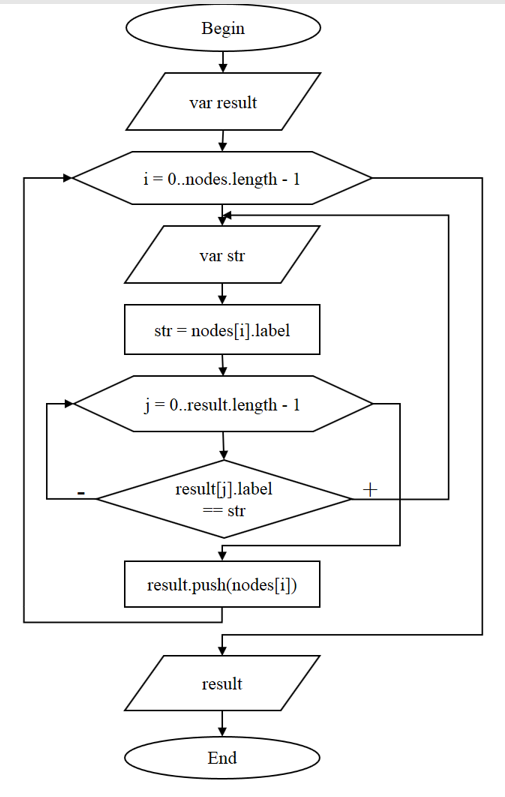
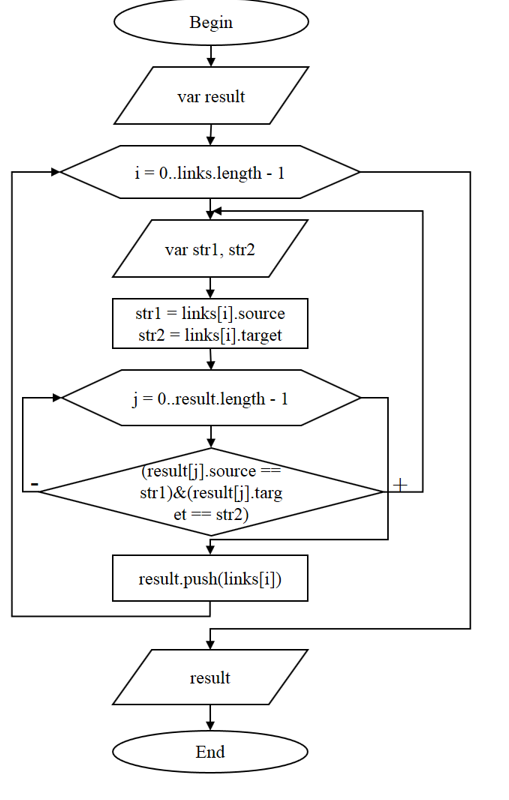


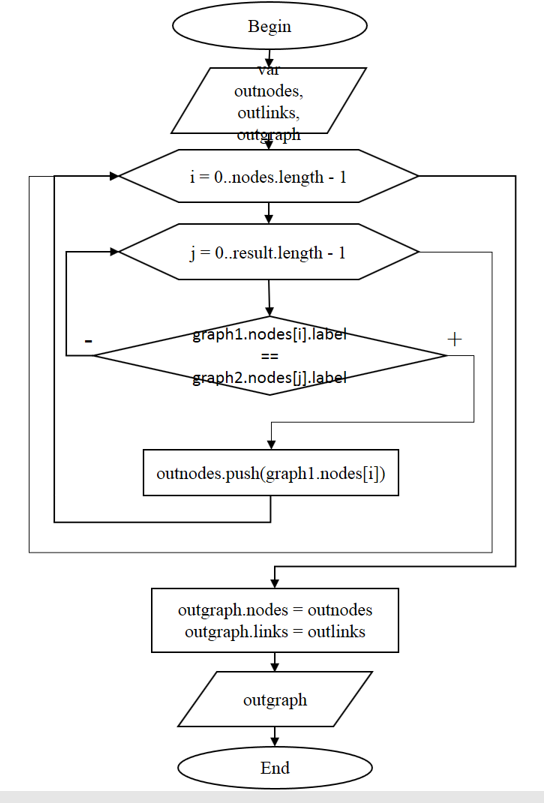
Схема работы функции uniqueLinks(links)



Исходный код функций и скриншоты программы приведены в Приложении №2

* + 1. **Операция пересечения.**

Схема работы функции, выполняющей пересечение двух онтологий graph1 и graph2



Исходный код функций и скриншоты программы приведены в Приложении №3

1. **Заключение**

В настоящей учебно-исследовательской работе был рассмотрен онтологический способ представления документальной информации.

Было определено понятие онтологий и четырех операций над ними:

– Объединение

– Пересечение

– Масштабирование

– Проекция

Была сформирована база данных. Для этого:

– были выделены свойства графа, необходимые для выполнения операций над онтологиями;

– были изучены способы задания данных в БД;

– на основании первых двух пунктов была построена структура базы данных.

Также была выбрана одна из трех графических библиотек D3.js.

Были реализованы две из четырех операций над онтологиями.

В результате была создана программная система, визуализирующая данные в виде графа с возможностью выделения отдельной вершины, выделения нескольких вершин, объединения и пересечения нескольких графов, возможностью масштабирования изображения графа и возможностью «перетаскивания» вершин.

1. **Список используемых источников**

– Голицына О. Л., Максимов Н. В., Окропишина О. В., Строгонов В. И. Онтологический подход к идентификации информации в задачах документального поиска. М.: НТИ. 2012. Сер. 2.– № 5. – С. 1-9.

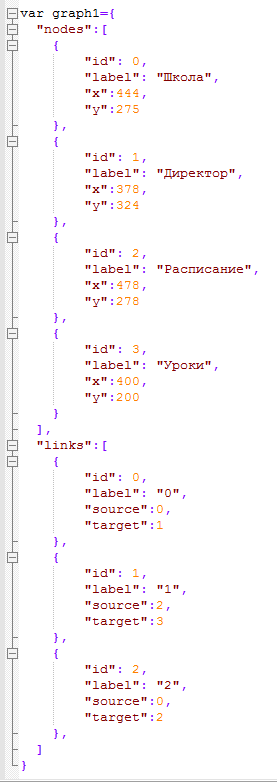
– О.Л. Голицына, Н.В. Максимов, О.В. Окропишина, В.И. Строгонов Онтологический подход к идентификации информации в задачах документального поиска: практическое применение. — М.: НТИ. 2013. Сер. 2.– № 3. – С. 1-8.

– Голицына О. Л., Максимов Н. В. Модели информационного поиска в контексте поисковых задач. М.: НТИ.,2011. Сер.2. - № 2. - С.1-12.

– [http://dic.academic.ru](https://www.antiplagiat.ru/go?to=MPjHV3iVvq6WQanFNDmu8dn7c1Yj8Plvk9HjKsCBUHXhwJDlB9iuArKhbe7UTsnOWqN8VwwX6owSoYMtTrt_LsKmuSfiVwX55sb-065PRw1o3_nD0)

1. **Приложение**

Приложение №1. Пример определения одного из графов в программной системе



Приложение №2. Исходный код функций, реализующих объединение двух графов и скриншоты работы программы

function union(graph1, graph2) {

var nodes = graph1.nodes.concat(graph2.nodes);

var links = graph1.links.concat(graph2.links);

var outgraph = {

nodes: uniqueNodes(nodes),

links: uniqueLinks(links)}

return (outgraph);

}

function uniqueNodes(nodes) {

var result = [];

nextInput:

for (var i = 0; i < nodes.length; i++) {

var str = nodes[i].label;

for (var j = 0; j < result.length; j++) {

if (result[j].label == str) continue nextInput;

}

result.push(nodes[i]);

}

return result;

}

function uniqueLinks(links) {

var result = [];

nextInput:

for (var i = 0; i < links.length; i++) {

var str1 = links[i].source;

var str2 = links[i].target;

for (var j = 0; j < result.length; j++) {

if ((result[j].source == str1)&(result[j].target == str2)) continue nextInput;

}

result.push(links[i]);

}

return result;

}

Исходные графы:

Результирующий граф:



Приложение №3. Исходный код функций, реализующих пересечение двух графов и скриншоты работы программы

function intersect(graph1, graph2) {

var outnodes = [];

var outlinks = [];

for (i = 0; i < graph1.nodes.length; i++)

for (j = 0; j < graph2.nodes.length; j++)

if (graph1.nodes[i].label == graph2.nodes[j].label)

{

outnodes.push(graph1.nodes[i]);

break;

}

for (i = 0; i < graph1.links.length; i++)

for (j = 0; j < graph2.links.length; j++)

if ((graph1.links[i].source == graph2.links[j].source)&(graph1.links[i].target == graph2.links[j].target))

{

outlinks.push(graph1.links[i]);

}

var outgraph = {

nodes: outnodes,

links: outlinks}

return (outgraph);

}

Исходные графы:

Результирующий граф:

