ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М. А. БОНЧ-БРУЕВИЧА»

Ф.В.Филиппов

ИНТЕЛЛЕКТУАЛИЗАЦИЯ УПРАВЛЕНИЯ ИНФОКОММУНИКАЦИОННЫМИ СИСТЕМАМИ И СЕТЯМИ

ПРАКТИКУМ

СПбГУТ)))
САНКТ-ПЕТЕРБУРГ

2017

ВВЕДЕНИЕ

Цель практических занятий состоит в освоении среды Apache Jena Fuseki, позволяющей эффективно формировать базы знаний на основе языка OWL для интеллектуализации управления поиском информации.

Apache Jena Fuseki - это сервер SPARQL [1]. Он может работать как служба операционной системы, как веб-приложение Java (WAR-файл), и как автономный сервер. Он обеспечивает безопасность (используя Apache Shiro) и имеет пользовательский интерфейс для мониторинга и администрирования сервера.

Он предоставляет протоколы SPARQL 1.1 для запросов и обновлений, а также протокол SPARQL Graph Store.

Fuseki тесно интегрирован с TDB для обеспечения надежного, устойчивого транзакционного уровня хранения и включает в себя текстовый запрос Jena и пространственный запрос Jena. Он может использоваться для обеспечения механизма протокола для других RDF-запросов и систем хранения.

1. ЯЗЫК OWL

Язык онтологий для Web – OWL (Web Ontology Language), так же как RDF и RDFS разработан для описания данных и метаданных, а также отношений между ними и предназначен для использования в компьютерной обработке данных семантического Web.

Язык OWL определен в группе из шести рекомендаций, принятой консорциумом W3 в феврале 2004 года.

Термин **онтология** заимствован из философии и обозначает науку, описывающую формы бытия и то, как они относятся между собой. В информатике под онтологией понимается всеобъемлющая и детальная формализация некоторой области знаний с помощью концептуальной схемы.

В Web онтология содержит описания классов, свойств и их реализацию в виде экземпляров классов. Формальная семантика OWL описывает, как с помощью такой онтологии сделать логические выводы, т.е. получить факты, которые не представлены в

онтологии напрямую, но следуют из ее семантики. Эти выводы могут быть основаны на одном документе или множестве распределенных документов, связанных между собой определенным образом.

Язык OWL предполагает открытость, т.е. описания ресурсов не ограничены единственным файлом или областью видимости. Класс OWL, первоначально определенный в одной онтологии, может быть расширен в других онтологиях, причем расширение не должно отменять первоначальное определение.

Для языка OWL определено следующее пространство имен:

http://www.w3.org/2002/07/owl#.

Обычно для этого пространства используется префикс **owl**.

1.1. Виды OWL

По степени реализации возможностей язык OWL имеет следующие три вида:

• OWL Lite (упрощенный OWL) – самый простой (для разработчиков как программных продуктов, так и приложений с использованием OWL) вид языка;

• OWL DL (OWL Description Logics – OWL с описательной логикой) – вид, разработанный для использования в бизнес-приложениях на основе логики первого порядка (описательной логики);

• OWL Full (полный OWL) – вид для высококвалифицированных разработчиков приложений, обеспечивающий полную реализацию выразительность и синтаксическую свободу RDF но без гарантий того, что компьютерная обработка данных привет к определенному результату (кроме того, этот вид очень труден для реализации разработчиками программных продуктов).

Виды OWL не являются независимыми языками, а построены на принципе расширения возможностей, т.е. вид OWL DL включает в себя вид OWL Lite с расширением существующих возможностей и некоторыми дополнительными возможностями, а вид OWL Full – это версия OWL DL с дополнительными возможностями. Соответственно правильное логическое заключение, сделанное с использованием OWL Lite, является правильным логическим заключением в OWL DL, а правильное логическое заключение, сделанное в OWL DL, является правильным логическим заключением в OWL Full.

Разработчики приложений, использующих OWL, должны решить, какой из видов лучше подходит к стоящим перед ним задачам. Выбор между OWL Lite и OWL DL зависит от того, насколько пользователям требуются более выразительные логические структуры для ограничений, предоставляемые OWL DL. Однако реализация в OWL DL требует большего времени как на разработку, так и на получение результата при обработке на компьютере. Выбор между OWL DL и OWL Full зависит, главным образом, от того, насколько пользователям требуются средства RDFS. Кроме того, при использовании OWL Full результат логического вывода требует большего времени вычисления, и результат не является гарантированным.

1.2. Структура документа OWL

Документ OWL – это документ на языке RDF/XML, который может содержать **заголовок OWL**, а также содержит определения классов, свойств и сведений о представителях классов. **Представители классов** (individuals) по терминологии OWL – это реализации (экземпляры) классов.

В качестве расширения файла с документом OWL можно использовать расширения **.owl** или **.rdf**. Значение MIME-типа данных для такого файла задается как

"application/rdf+xml".

Класс **owl:Ontology** используется для описания заголовка OWL, который в языке RDF/XML имеет следующий синтаксис:

```
<owl:Ontology rdf:about="ресурс">
    …
</ >
```

В атрибуте **rdf:about** задается наименование онтологии или ссылка на онтологию. Если значение атрибута – пустая строка (""), то наименованием онтологии служит базовый URI экземпляра класса **owl:Ontology**. Как правило, это URI документа, содержащего онтологию. Исключение является случай, когда в элементе **rdf:RDF** задан атрибут **xml:base**, явно устанавливающий базовый URI документа.

Классами, определяющими семантические свойства классов OWL, являются **owl:AnnotationProperty** и **owl:OntologyProperty**. Оба этих класса являются подклассами класса **rdf:Property** (см. 1.3.1).

В элементе **owl:Ontology** могут быть заданы следующие информационные свойства RDFS (см. 1.3.1): **rdfs:label** и **rdfs:comment**, а также вспомогательные свойства RDFS (см. 1.3.7): **rdfs:seeAlso** и **rdfs:isDefinedBy**. Эти свойства одновременно являются экземплярами класса **owl:AnnotationProperty**.

Также в элементе **owl:Ontology** могут быть заданы следующие свойства OWL: **owl:imports**, **owl:versionInfo**, **owl:priorVersion**, **owl:incompatibleWith** и **owl:backwardCompatibleWith**.

Свойство **owl:imports** является экземпляром класса **owl:OntologyProperty**. Он определяет URI документа, импортируемого в данный документ. В результате импорта в данный документ будет вставлено содержимое импортируемого документа. Импорт документов может быть вложенным, т.е. импортируемый документ, в свою очередь, может содержать другой импортируемый документ.

Значениями свойств **rdfs:range** и **rdfs:domain** для свойства **owl:imports** является **owl:Ontology**.

Свойство **owl:versionInfo** является экземпляром класса **owl:AnnotationProperty**. Обычно его объектом является строка, содержащая сведения о версии данной онтологии.

Свойство **owl:priorVersion** является экземпляром класса **owl:OntologyProperty**. Это свойство содержит ссылку на другую онтологию, которая таким образом рассматривается как предыдущая версия данной онтологии.

Значениями свойств **rdfs:range** и **rdfs:domain** для свойства **owl:priorVersion** является **owl:Ontology**.

Свойство **owl:incompatibleWith** является экземпляром класса **owl:OntologyProperty**. Это свойство содержит ссылку на другую онтологию, которая таким образом рассматривается как предыдущая версия данной онтологии, несовместимая с данной онтологией.

Значениями свойств **rdfs:range** и **rdfs:domain** для свойства **owl:incompatibleWith** является **owl:Ontology**.

Свойство **owl:backwardCompatibleWith** является экземпляром класса **owl:OntologyProperty**. Это свойство содержит ссылку на другую онтологию, которая таким образом рассматривается как предыдущая версия данной онтологии, совместимая с данной онтологией.

Значениями свойств **rdfs:range** и **rdfs:domain** для свойства **owl:backwardCompatibleWith** является **owl:Ontology**.

Если в классе **owl:Ontology** задано свойство **owl:backwardCompatibleWith**, то документ OWL может содержать классы **owl:DeprecatedClass** и **owl:DeprecatedProperty**, описывающие классы и

свойства предыдущей версии, которые в данной версии отменены или заменены другими классами и свойствами.

Класс **owl:DeprecatedClass** является подклассом класса **rdfs:Class**, а класс **owl:DeprecatedProperty** – подклассом класса **rdf:Property**.

Пример 1.1. Документ OWL с заголовком на языке RDF/XML:

```
<?xml version="1.0"?>
     <!-- Описание сущностей документа -->
<!DOCTYPE rdf:RDF [
     <!ENTITY rdf
     "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
     <!ENTITY rdfs
     "http://www.w3.org/2000/01/rdf-schema#" >
     <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#" >
     <!ENTITY owl  "http://www.w3.org/2002/07/owl#" >
   ]>

<rdf:RDF
          <!-- Задание префиксов пространств имен -->
     xmlns:owl ="&owl;"
     xmlns:rdf ="&rdf;"
     xmlns:rdfs="&rdfs;"
          <!-- Задание базового URI -->
     xml:base  ="http://myOntology.edu/version1-1"
>

     <owl:Ontology rdf:about="">
               <!-- Импортирование документа версии 1.0 -->
          <owl:imports rdf:resource=
          "http://myOntology.edu/version1-0"/>
               <!-- Ссылка на документацию -->
          <rdfs:isDefinedBy rdf:resource=
          "http://myOntology.edu/version1-1/doc"/>
          <rdfs:comment>
               <!-- Комментарий -->
          Версия 1.1 онтологии myOntology</rdfs:comment>
               <!-- Описание версии -->
          <owl:versionInfo>myOntology, v1.1</versionInfo>
               <!-- Ссылка на предыдущую версию -->
          <owl:priorVersion rdf:resource=
          "http://myOntology.edu/version1-0"/>
               <!-- Несовместимость с версией 0.9 -->
          <owl:incompatibleWith rdf:resource=
          "http://myOntology.edu/version0-9"/>
               <!-- Обратная совместимость с версией 1.0 -->
          < owl:backwardCompatibleWith rdf:resource=
          "http://myOntology.edu/version1-0"/>
     </Ontology>
     …
          <!-- Описание отмененного класса Student -->
     <owl:DeprecatedClass rdf:ID="Student">
               <!-- Комментарий -->
          <rdfs:comment>Вместо класса Student рекомендуется
```

```
            использовать класс Learner</rdfs:comment>
                <!-- Ссылка на эквивалентный класс
                (см. ниже описание owl:equivalentClass) -->
            <owl:equivalentClass rdf:resource="#Learner"/>
        </owl:DeprecatedClass>
        …
        <owl:Class rdf:ID="Learner">
            <!-- Описание класса Learner -->
        </owl:Class>
            <!-- Описание отмененного свойства hasParents -->
        <owl:DeprecatedProperty rdf:ID="hasParents">
                <!-- Комментарий -->
            <rdfs:comment>Свойство hasParents
            в версии 1.1 отменено</rdfs:comment>
        </owl:DeprecatedProperty>
        …
    </rdf:RDF>
```

1.3. Классы OWL

Как и классы RDF, классы OWL обеспечивают группирование ресурсов со сходными характеристиками. Каждый класс связан с набором представителей класса, называемым **расширением класса**. Представители класса в расширении класса называются **экземплярами класса**. Каждый класс имеет некоторый содержательный смысл, который связан, но не эквивалентен расширению класса, т.е. два класса могут имеет одинаковые расширения, но являться разными классами.

В языках OWL Lite и OWL DL представитель класса не может в то же время классом. Язык OWL Full может использовать возможность RDFS: класс может функционировать как экземпляр другого класса.

Классы OWL описываются с помощью **описаний класса**, которые затем комбинируются в **аксиомы класса**.

1.3.1. Описания класса

В языке OWL определены шесть типов описаний классов:

- с помощью идентификатора класса;
- с помощью перечисления представителей класса;
- с помощью ограничения свойств;
- с помощью пересечения двух и более описаний классов;
- с помощью объединения двух и более описаний классов;
- с помощью дополнения описания класса.

При использовании первого типа определения класс задается с определенным именем. В остальных типах класс задается как пустой узел со свойством **`rdf:type`**, чье значение равно **`owl:Class`**.

1.3.1.1. Описание класса с помощью идентификатора

Описание класса с помощью идентификатора в нотации N3 задается следующим образом:

префикс`:`*имя-класса* **`rdf:type owl:Class`**,

где *префикс* – это префикс пространство имен, в котором определен новый класс с именем *имя-класса*.

В RDF/XML класс задается с помощью элемента **`owl:Class`** с атрибутом **`rdf:ID`**.

Класс **owl:Class** является подклассом класса **rdfs:Class**. Класс **owl:Class** в OWL Lite и OWL DL имеет одно существенное ограничение по сравнению с классом **rdfs:Class**: класс не может одновременно являться экземпляром своего класса. В OWL Full этого ограничения нет, и оба класса считаются эквивалентными.

Два класса с идентификаторами **Thing** и **Nothing** в языке OWL являются предопределенными.

Расширением класса для **owl:Thing** является набор всех представителей этого класса, а расширением класса **owl:Nothing** является пустой набор. Соответственно каждый класс OWL является подклассом **owl:Thing**, а класс **owl:Nothing** является подклассом каждого класса.

Пример 1.2. Объявление класса OWL в языке RDF/XML:

Объявление класса OWL с именем **Office**:

```
<owl:Class rdf:ID="Office"/>.
```

1.3.1.2. Описание класса с помощью перечисления

Описание класса с помощью перечисления выполняется с помощью свойства **owl:oneOf**. Значением этого свойства является список представителей класса (его экземпляров).

Значениями свойств **rdfs:range** и **rdfs:domain** для свойства **owl:oneOf** являются соответственно **rdf:List** и **owl:Class**.

Список представителей класса задается в RDF/XML с помощью атрибута **rdf:parseType="Collection"**.

Объявить класс по перечислению можно только в языках OWL DL и OWL Full.

Пример 1.3. Объявление класса OWL по перечислению в языке RDF/XML:

Объявление класса OWL, представителями которого являются цвета RGB:

```
<owl:Class>
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Red"/>
    <owl:Thing rdf:about="#Green"/>
    <owl:Thing rdf:about="#Blue"/>
  </owl:oneOf>
</owl:Class>.
```

1.3.1.3. Описание класса с помощью ограничения свойств

Под описанием класса с помощью ограничения свойств понимается описание безымянного класса, все представители которого удовлетворяют заданным ограничениям.

Ограничения определяются с помощью класса **owl:Restriction**, являющегося подклассом класса **owl:Class**.

Значение ограничения (ресурс или литерал) задаются с помощью свойства **owl:onProperty**. Значениями свойств **rdfs:range** и **rdfs:domain** для этого свойства являются соответственно **rdf:Property** и **owl:Restriction**.

Для ограничения может быть задано только одно свойство **owl:onProperty**.

В RDF/XML ограничение задается с помощью элемента **owl:Restriction**, в который вложен элемент **owl:onProperty**.

В OWL определены два вида ограничения свойств: ограничения по значению (value constraints) и ограничения по мощности (cardinality constraints).

Ограничения по значению задают диапазон свойства, используемого в описании данного класса. В OWL определены следующие свойства ограничения по значению: **owl:allValuesFrom**, **owl:someValuesFrom**, и **owl:hasValue**.

Значениями свойств **rdfs:range** и **rdfs:domain** для этих свойств являются соответственно **rdf:Class** и **owl:Restriction**.

В OWL Lite единственным типом описания класса, допустимым в качестве объекта в свойствах ограничения по значению является имя класса.

Свойство owl:allValuesFrom связывает ограничиваемый класс либо с описанием класса, либо с диапазоном данных. Это свойство используется для описания класса всех представителей, для которых все значения свойства либо описания классов, либо значения данных в заданном диапазоне.

Пример 1.4. Использование свойства owl:allValuesFrom в языке RDF/XML:

Описание ограничения класса **hasColor**, представителями которого являются только значения класса **RGBColor** (цвета RGB):

```
<owl:Restriction>
     <owl:onProperty rdf:resource="#hasColor"/>
     <owl:allValuesFrom rdf:resource="#RGBColor"/>
</owl:Restriction>.
```

Свойство owl:someValuesFrom связывает ограничиваемый класс либо с описанием класса, либо с диапазоном данных. Это свойство используется для описания класса всех представителей, для которых по крайней мере одно значение свойства – экземпляр либо описания класса, либо значения данных в заданном диапазоне.

Пример 1.5. Использование свойства owl:someValuesFrom в языке RDF/XML:

Описание ограничения класса **hasColor**, среди представителей которого имеется, по крайней мере, один представитель для белого цвета (**whiteColor**):

```
<owl:Restriction>
     <owl:onProperty rdf:resource="#hasColor"/>
     <owl:allValuesFrom rdf:resource="#whiteColor"/>
</owl:Restriction>.
```

Свойство owl:hasValue связывает ограничиваемый класс с заданным значением, которое является либо представителем класса, либо значением данных. Это свойство используется для описания класса всех представителей, для которых по крайней мере одно значение свойства – экземпляр либо описания класса, либо значения данных в заданном диапазоне.

Пример 1.6. Использование свойства owl:hasValue в языке RDF/XML:

Описание ограничения класса **hasColor**, среди представителей которого имеется ссылка на представитель класса для черного цвета (**blackColor**):

```
<owl:Restriction>
     <owl:onProperty rdf:resource="#hasColor"/>
     <owl:hasValue rdf:resource="#blackColor"/>
</owl:Restriction>.
```

Ограничение по мощности задает количество значений, которое может иметь свойство. В OWL определены следующие свойства ограничения по мощности: **owl:maxCardinality**, **owl:minCardinality**, и **owl:Cardinality**.

Значениями свойств **rdfs:range** и **rdfs:domain** для этих свойств являются соответственно **xsd:nonNegativeInteger** и **owl:Restriction**.

В OWL Lite свойства ограничения по мощности могут иметь только значения **"0"** и **"1"**.

Свойство `owl:maxCardinality` связывает ограничиваемый класс со значением данных, принадлежащим к пространству значений типа `nonNegativeInteger` схемы XML. Свойство описывает класс всех представителей, который имеет самое большее *N* различных значений, где *N* – значение ограничения мощности.

В RDF/XML ограничение задается как значение элемента `owl:maxCardinality` с атрибутом `rdf:datatype`, который имеет значение `"&xsd;nonNegativeInteger"`.

Пример 1.7. Использование свойства owl:maxCardinality в языке RDF/XML:

Описание ограничения класса `SmallOfficeStaff`, который имеет самое большее 10 представителей:

```
<owl:Restriction>
     <owl:onProperty rdf:resource="#SmallOfficeStaff"/>
     <owl:maxCardinality rdf:datatype=
     "&xsd;nonNegativeInteger">10</owl:maxCardinality>
</owl:Restriction>.
```

Свойство `owl:minCardinality` связывает ограничиваемый класс со значением данных, принадлежащим к пространству значений типа `nonNegativeInteger` схемы XML. Свойство описывает класс всех представителей, который имеет самое меньшее *N* различных значений, где *N* – значение ограничения мощности.

В RDF/XML ограничение задается так же, как для элемента `owl:maxCardinality`.

Пример 1.8. Использование свойства owl:minCardinality в языке RDF/XML:

Описание ограничения класса `BigOfficeStaff`, который имеет самое меньшее 100 представителей:

```
<owl:Restriction>
     <owl:onProperty rdf:resource="#SmallOfficeStaff"/>
     <owl:minCardinality rdf:datatype=
     "&xsd;nonNegativeInteger">100</owl:minCardinality>
</owl:Restriction>.
```

Свойство `owl:cardinality` связывает ограничиваемый класс со значением данных, принадлежащим к пространству значений типа `nonNegativeInteger` схемы XML. Свойство описывает класс всех представителей, который имеет точно *N* различных значений, где *N* – значение ограничения мощности.

В RDF/XML ограничение задается так же, как для элемента `owl:maxCardinality`.

Пример 1.9. Использование свойства owl:cardinality в языке RDF/XML:

Описание ограничения класса `SomeOfficeStaff`, который имеет точно 8 представителей:

```
<owl:Restriction>
     <owl:onProperty rdf:resource="#SmallOfficeStaff"/>
     <owl:cardinality rdf:datatype=
     "&xsd;nonNegativeInteger">8</owl:cardinality>
</owl:Restriction>.
```

1.3.1.4. Описание класса с помощью пересечения

Описание класса с помощью пересечения (операция **AND** – логическое **И**) выполняется с помощью свойства **owl:intersectionOf**, связывающего класс со списком описаний классов. Это свойство описывает безымянный класс, в котором расширение содержит только те представители, которые содержатся во всех расширениях в описании классов в списке.

Значениями свойств **rdfs:range** и **rdfs:domain** для свойства **owl:intersectionOf** являются соответственно **rdf:List** и **owl:Class**.

В OWL Lite значения свойство **owl:intersectionOf** могут быть только идентификаторами класса и/или ограничениями свойств.

Пример 1.10. Использование свойства owl:intersectionOf в языке RDF/XML:

Описание ограничений, являющихся пересечением двух коллекций цветов, для класса:

```
<owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class>
                    <!-- Коллекция из двух элементов:
                    черного и красного цвета -->
               <owl:oneOf rdf:parseType="Collection">
                    <owl:Thing rdf:about="#blackColor"/>
                    <owl:Thing rdf:about="#redColor"/>
               </owl:oneOf>
          </owl:Class>
          <owl:Class>
                    <!-- Коллекция из четырех элементов:
                    красного, зеленого, синего
                    и белого цветов -->
               <owl:oneOf rdf:parseType="Collection">
                    <owl:Thing rdf:about="#redColor"/>
                    <owl:Thing rdf:about="#greenColor"/>
                    <owl:Thing rdf:about="#blueColor"/>
                    <owl:Thing rdf:about="#whiteColor"/>
               </owl:oneOf>
          </owl:Class>
     </owl:intersectionOf>
</owl:Class>.
```

Этот класс будет иметь только одного представителя – **redColor** (красный цвет), поскольку только этот представитель содержится в обоих списках.

1.3.1.5. Описание класса с помощью объединения

Описание класса с помощью объединения (операция **OR** – логическое **ИЛИ**) выполняется с помощью свойства **owl:unionOf**, связывающего класс со списком описаний классов. Это свойство описывает безымянный класс, в котором расширение содержит все представители, которые содержатся хотя бы в одном из расширений класса в описании классов в списке.

Значениями свойств **rdfs:range** и **rdfs:domain** для свойства **owl:unionOf** являются соответственно **rdf:List** и **owl:Class**.

Свойство **owl:unionOf** можно использовать только в OWL DL и OWL Full.

Пример 1.11. Использование свойства owl:unionOf в языке RDF/XML:

Описание ограничений, являющихся объединением двух коллекций цветов, для класса:

```
<owl:Class>
     <owl:unionOf rdf:parseType="Collection">
          <owl:Class>
                    <!-- Коллекция из двух элементов:
                    черного и красного цвета -->
               <owl:oneOf rdf:parseType="Collection">
                    <owl:Thing rdf:about="#blackColor"/>
                    <owl:Thing rdf:about="#redColor"/>
               </owl:oneOf>
          </owl:Class>
          <owl:Class>
                    <!-- Коллекция из четырех элементов:
                    красного, зеленого, синего
                    и белого цветов -->
               <owl:oneOf rdf:parseType="Collection">
                    <owl:Thing rdf:about="#redColor"/>
                    <owl:Thing rdf:about="#greenColor"/>
                    <owl:Thing rdf:about="#blueColor"/>
                    <owl:Thing rdf:about="#whiteColor"/>
               </owl:oneOf>
          </owl:Class>
     </owl:unionOf>
</owl:Class>.
```

Этот класс содержит пять представителей: **blackColor** (черный цвет), **redColor** (красный цвет), **greenColor** (зеленый цвет), **blueColor** (синий цвет) и **whiteColor** (белый цвет), т.е. представителями класса является сумма представителей обоих списков (представитель **redColor** есть в обоих списках, но в суммарном списке он будет представлен только один раз).

1.3.1.6. Описание класса с помощью дополнения

Описание класса с помощью дополнения (операция **NOT** – логическое **НЕ**) выполняется с помощью свойства **owl:complementOf**, описывающего класс, для которого содержит только те представители, которые не принадлежат к расширению класса, являющегося объектом предложения.

Значениями свойств **rdfs:range** и **rdfs:domain** для свойства **owl:complementOf** являются соответственно **rdf:List** и **owl:Class**.

Свойство **owl:complementOf** можно использовать только в OWL DL и OWL Full.

Пример 1.12. Использование свойства owl:complementOf в языке RDF/XML:

Описание ограничения, являющегося дополнением для класса:

```
<owl:Class>
  <owl:complementOf>
    <owl:Class rdf:about="#greyColor"/>
  </owl:complementOf>
</owl:Class>.
```

Расширение этого класса содержит всех тех представителей, которые не принадлежат классу **greyColor**.

1.3.2. Аксиомы классов

Описания классов образуют компоненты для определения классов с помощью **аксиом классов**. Простейшей формой аксиомы класса является описание класса с помощью идентификатора, однако обычно аксиомы содержат дополнительные компоненты, задающие необходимые и/или достаточные характеристики классов.

Для комбинирования описания класса в аксиому класса используются следующие свойства: `rdfs:subClassOf`, `owl:equivalentClass` и `owl:disjointWith`.

Свойство `rdfs:subClassOf`, определенное в RDFS, задает, что расширение класса в описании класса является подмножеством расширения класса в описании другого класса.

Пример 1.13. Использование свойства rdfs:subClassOf в аксиоме класса:

```
<owl:Class rdf:ID="RGBColor">
          <!-- Класс RGBColor является подклассом
          класса Color -->
     <rdfs:subClassOf rdf:resource="#Color"/>
          <!-- Ограничения на свойство hasValue
          класса RGBColor: один из трех цветов:
          красный, зеленый или синий -->
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#hasValue"/>
               <owl:someValuesFrom>
                    <owl:Class>
                         <owl:oneOf rdf:parseType=
                         "Collection">
                              <owl:Thing rdf:about=
                              "#RedColor"/>
                              <owl:Thing rdf:about=
                              "#GreenColor"/>
                              <owl:Thing rdf:about=
                              "#BlueColor"/>
                         </owl:oneOf>
                    </owl:Class>
               </owl:someValuesFrom>
          </owl:Restriction>
     </rdfs:subClassOf>
          <!-- Класс RGBColor является дополнением
          к классу CMYKColor -->
     <rdfs:subClassOf>
          <owl:Class>
               <owl:complementOf rdf:resource="#CMYKColor"/>
          </owl:Class>
     </rdfs:subClassOf>
</owl:Class>
```

Свойство `owl:equivalentClass` связывает описание данного класса с описанием другого класса. Использование этого свойства в аксиоме означает, что оба класса имеют одинаковое расширение, т.е. оба расширения класса содержат одинаковый набор представителей.

Значением свойств `rdfs:range` и `rdfs:domain` для свойства `owl:equivalentClass` является `owl:Class`.

Пример 1.14. Использование свойства owl:equivalentClass в аксиоме класса:

1. Объявление именованного класса **BaseColor** эквивалентным именованному классу **RGBColor**:

```
<owl:Class rdf:about="#BaseColor">
     <owl:equivalentClass rdf:resource="#RGBColor"/>
</owl:Class>
```

2. Объявление именованного класса **CMYKColor** эквивалентным безымянному вложенному классу, содержащему расширение из четырех представителей: **CyanColor** (циановый цвет), **MagentaColor** (фиолетовый цвет), **YellowColor** (желтый цвет) и **BlackColor** (черный цвет):

```
<owl:Class rdf:ID="CMYKColor">
     <owl:equivalentClass>
          <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                    <owl:Thing rdf:about="#CyanColor"/>
                    <owl:Thing rdf:about="#MagentaColor"/>
                    <owl:Thing rdf:about="#YellowColor"/>
                    <owl:Thing rdf:about="#BlackColor"/>
               </owl:oneOf>
          </owl:Class>
     </owl:equivalentClass>
</owl:Class>
```

3. В RDF/XML можно задать эквивалентность определяемого класса и вложенного класса в сокращенной форме (без использования элемента **owl:equivalentClass**). Предыдущий пример при использовании сокращенной формы будет иметь следующий вид:

```
<owl:Class rdf:ID="CMYKColor">
     <owl:oneOf rdf:parseType="Collection">
          <owl:Thing rdf:about="#CyanColor"/>
          <owl:Thing rdf:about="#MagentaColor"/>
          <owl:Thing rdf:about="#YellowColor"/>
          <owl:Thing rdf:about="#BlackColor"/>
      </owl:oneOf>
</owl:Class>
```

Свойство **owl:disjointWith** связывает описание данного класса с описанием другого класса. Использование этого свойства в аксиоме означает, что расширения двух классов не имеют общих представителей.

Значением свойств **rdfs:range** и **rdfs:domain** для свойства **owl:disjointWith** является **owl:Class**.

Пример 1.15. Использование свойства owl:disjointWith в аксиоме класса:

Объявление об отсутствии общих представителей в классах **CMYKColor** и **RGBColor**:

```
<owl:Class rdf:about="#CMYKColor">
     <owl:disjointWith rdf:resource="#RGBColor"/>
</owl:Class>.
```

1.4. Свойства OWL

В языке OWL определены следующие категории свойств:

- свойства онтологий (ontology properties);
- свойства аннотаций (annotation properties);
- свойства объектов (object properties);
- свойства типизированных данных (datatype properties).

Классами, определяющими свойства онтологий и аннотаций, являются **owl:AnnotationProperty** и **owl:OntologyProperty**, рассмотренные в разд. 1.2.

Свойства объектов в OWL определяются как экземпляры класса **owl:ObjectProperty**, а свойства типизированных данных – как экземпляры класса **owl:DatatypeProperty**. Оба этих класса являются подклассами класса **rdf:Property**.

В языке OWL Full свойства объектов и свойства типизированных данных не являются взаимоисключающими, поскольку значения данных можно рассматривать как представления. Поэтому в OWL Full класс **owl:ObjectProperty** эквивалентен классу **rdf:Property**.

Аксиома свойства определяет характеристики свойства.

В простейшем случае аксиома свойства просто задает существование свойства, например

```
<owl:ObjectProperty rdf:ID="hasValue"/>.
```

Однако чаще аксиомы свойства задают дополнительные характеристики свойств. В языке OWL поддерживаются следующие конструкции для характеристик свойств:

- свойства RDFS;
- отношения к другим свойствам;
- глобальные ограничения мощности;
- характеристики логических свойств.

В OWL DL субъект и объект подсвойства должны быть оба либо свойствами типизированных данных, либо свойствами объектов.

В OWL Lite значением свойства **rdfs:domain** и **rdfs:range** должен быть только идентификатор класса.

Остальные конструкции для характеристик свойств рассматриваются ниже.

1.4.1. Отношения к другим свойствам

Для задания отношения к другим свойствам в языке OWL используются свойства **owl:equivalentProperty** и **owl:inverseOf**.

Свойство owl:equivalentProperty используется для указания того, что два свойства имеют одинаковое расширение.

Значением свойств **rdfs:range** и **rdfs:domain** для свойства **owl:equivalentProperty** является **rdf:Property**.

Пример 1.16. Использование свойства owl:equivalentProperty:

Объявление об эквивалентности свойств **hasColorValue** и **hasValue**:

```
<owl:ObjectProperty rdf:ID="hasColorValue">
     < owl:equivalentProperty rdf:resource="#hasValue"/>
</owl:ObjectProperty>.
```

Свойство имеет направление – от домена к диапазону. Иногда необходимо определить отношения в обоих направления. Например, для отношения *человек владеет автомашиной* обратным является отношение *автомашина принадлежит человеку*.

Свойство `owl:inverseOf` формирует обратное отношение для свойства или, более строго, аксиома

P1 `owl:inverseOf` *P2*

утверждает, что для каждой пары (*x*,*y*) в расширении *P1*существует пара (*y*,*x*) в расширении *P2*, и наоборот, т.е. свойство `owl:inverseOf` является симметричным

Значением свойств `rdfs:range` и `rdfs:domain` для свойства `owl:inverseOf` является `owl:ObjectProperty`.

Пример 1.17. Использование свойства owl:inverseOf:

Объявление об том, что у свойства `hasDescendant` (имеет потомка) существует обратное свойство `hasAncestor` (имеет предка):

```
<owl:ObjectProperty rdf:ID="hasDescendant">
      <owl:inverseOf rdf:resource="#hasAncestor"/>
</owl:ObjectProperty>.
```

1.4.2. Глобальные ограничения мощности

Глобальные ограничения мощности задаются с помощью классов `owl:FunctionalProperty` и `owl:InverseFunctionalProperty`.

Эти ограничения называются глобальными, потому что, к какому бы классу не применялось свойство, ограничения будут выполнены. В тоже время ограничения, рассмотренные в разд. 1.3.1.3 применимы только классам, в описания которых они включены.

Класс `owl:FunctionalProperty` определяет так называемое **функциональное свойство**, т.е. свойство, которое имеет одно (уникальное) значение *y* для каждого значения *x*, т.е. нет такой пары значений *y*1 и *y*2, таких, что пары (*x*,*y*1) и (*x*,*y*2) обе являются экземплярами свойства. Функциональным свойством может быть как свойство объекта, так и свойство типизированного данного. Класс `owl:FunctionalProperty` является специальным подклассом класса `rdf:Property`.

Пример 1.18. Использование класса owl:FunctionalProperty:

1. Объявление об том, что свойство объекта `fillColor` (цвет фона) является функциональным, т.е. для ресурса свойство может принимать одно из значений, определенных в классе `RGBColor`:

```
<owl:ObjectProperty rdf:ID="fillColor">
     <rdf:type    rdf:resource="&owl;FunctionalProperty"/>
     <rdfs:domain rdf:resource="#Shape"/>
     <rdfs:range  rdf:resource="#RGBColor"/>
</owl:ObjectProperty>.
```

2. Более краткая запись предыдущего примера:

```
<owl:ObjectProperty rdf:ID="fillColor">
     <rdfs:domain rdf:resource="#Shape" />
     <rdfs:range  rdf:resource="#RGBColor" />
</owl:ObjectProperty>
…
<owl:FunctionalProperty rdf:about="#fillColor" />.
```

Если свойство объявляется обратно функциональным, то объект в утверждении для свойства однозначно определяет субъект (некоторое представление класса). Более точно, если задано утверждение, что *P* – это обратное функциональное свойство, оно предполагает, что значение *y* может быть значением *P* для одного экземпляра *x*, т.е. не

может быть двух различных экземпляров $x1$ и $x2$ таких, что обе пары $(x1,y)$ и $(x2,y)$ являются экземплярами P. Обратное функциональное свойство соответствует понятию ключа в базе данных.

Аксиома обратного функционального свойства задается объявлением свойства как экземпляра **класса** `owl:InverseFunctionalProperty`, который является подклассом класса `owl:ObjectProperty`. По своему определению обратными функциональными свойствами могут быть только свойства объектов.

Поскольку в OWL Full свойства типизированных данных являются подклассами свойств объектов, обратное функциональное свойство может быть определено и для свойств типизированных данных. Такое определение недопустимо в OWL Lite и OWL DL.

Пример 1.19. Использование класса owl:InverseFunctionalProperty:

Объявление об том, что каждый объект предложений `birthRegion` (представитель класса `Person`) может однозначно идентифицировать субъект (представитель класса `Region`):

```
<owl:InverseFunctionalProperty rdf:ID="birthRegion">
     <rdfs:domain rdf:resource="#Region"/>
     <rdfs:range rdf:resource="#Person"/>
</owl:InverseFunctionalProperty>.
```

1.4.3. Характеристики логических свойств

Характеристики логических свойств (транзитивное или симметричное свойство) задаются с помощью классов `owl:TransitiveProperty` и `owl:SymmetricProperty`.

Если свойство P задается как **транзитивное**, это означает, что если пара (x,y) является экземпляром P и пара (y,z) также является экземпляром P, то пара (x,z) – тоже экземпляр P.

Аксиома транзитивного свойства задается объявлением свойства как экземпляра **класса** `owl:TransitiveProperty`, который является подклассом класса `owl:ObjectProperty`.

В OWL DL для транзитивного свойства нельзя задавать ограничения мощности (ни локальные, ни глобальные) ни для свойств или их суперсвойств, ни для обратных свойств или их суперсвойств.

Пример 1.20. Использование класса owl:TransitiveProperty:

1. Объявление транзитивного свойства `descendantOf` (потомок):

```
<owl:TransitiveProperty rdf:ID="descendantOf">
     <rdfs:domain rdf:resource="#Person"/>
     <rdfs:range  rdf:resource="#Person"/>
</owl:TransitiveProperty>.
```

Пусть `Иванов Иван Петрович`, `Иванов Петр Александрович` и `Иванов Александр Николаевич` являются представителями класса `Person`. Тогда, если для `Иванова Петра Александровича` значение свойства `descendantOf` (потомок) равно `Иванов Иван Петрович`, и для `Иванова Александра Николаевича` значение свойства `descendantOf` (потомок) равно `Иванов Петр Александрович`, то тогда, в силу транзитивности свойства `descendantOf` при семантической обработке документа можно сделать вывод, что `Иванов Иван Петрович` является потомком `Иванова Александра Николаевича`.

2. Поскольку `owl:TransitiveProperty` является подклассом `owl:ObjectProperty`, предыдущий пример можно записать в следующей эквивалентной форме:

```
<owl:ObjectProperty rdf:ID="descendantOf">
     <rdf:type rdf:resource="&owl;TransitiveProperty"/>
     <rdfs:domain rdf:resource="#Person"/>
     <rdfs:range  rdf:resource="#Person"/>
</owl:ObjectProperty>.
```

Симметричное свойство – это свойство, для которого предполагается, что если пара (x,y) является экземпляром P, тогда пара (y,x) – также экземпляр P.

Аксиома симметричного свойства задается объявлением свойства как экземпляра **класса** `owl:SymmetricProperty`, который является подклассом класса `owl:ObjectProperty`.

Для симметричного свойства значения свойств должны быть одинаковы.

Пример 1.21. Использование класса owl:SymmetricProperty:

1. Объявление транзитивного свойства `brotherOf` (брат):

```
<owl:SymmetricProperty rdf:ID="brotherOf">
     <rdfs:domain rdf:resource="#Person"/>
     <rdfs:range  rdf:resource="#Person"/>
</owl:SymmetricProperty>.
```

Пусть для `Иванов Иван Петрович` значение свойства `brotherOf` равно `Иванов Константин Петрович`, т.е. `Иванов Иван Петрович` является братом `Иванова Константина Петровича`. Тогда, в силу симметричности свойства `brotherOf` при семантической обработке документа можно сделать вывод, что и `Иванов Константин Петрович` является братом `Иванова Ивана Петровича`.

1.5. Представители классов в OWL

Представители классов в языке OWL определяются с помощью специальных аксиом, также называемых **фактами**. В OWL для представителей определены два типа фактов:

- факты о принадлежности к классу и значениях свойств;
- факты о тождественности представителей.

Факты о принадлежности к классу и значении свойств просто задают имя представителя, класс, экземпляром которого является представитель и набор значений свойств представителя, аналогично заданию экземпляров класса в RDFS.

Представитель класса не обязательно может иметь имя, он может быть и безымянным. В этом случае в элементе определения представителя не задается атрибут `rdf:ID`.

Факты о тождественности представителей задают утверждения относительно того, являются ли какие-либо представителя тождественными друг другу, т.е. относятся ли они к одному и тому же ресурсу.

В языке OWL определены три конструкции для задания факта тождественности представителей: `owl:sameAs`, `owl:differentFrom` и `owl:AllDifferent`.

Свойство `owl:sameAs` связывает двух представителей отношением тождественности, т.е. ссылки URI в них указывают на один и тот же ресурс.

Значением свойств `rdfs:range` и `rdfs:domain` для свойства `owl:sameAs` является `rdf:Thing`.

Понятие тождественности отличается от понятия эквивалентности следующим образом. Эквивалентность означает, что расширения двух классов одинаковы, но не подразумевает, что эти классы указывают на один и тот же ресурс.

В OWL Full класс рассматривается одновременно и как свой представитель, поэтому в нем можно задавать тождественность классов, т.е. два класса считаются

тождественными, если они реализованы по-разному, но имеют одинаковое содержательное значение.

Пример 1.22. Использование свойства owl:sameAs:

1. Объявление тождественности ссылки на ресурс **Kiev** и ссылки на ресурс **CapitalOfUkraine**, поскольку Киев является столицей Украины:

```
<rdf:Description rdf:about="#Kiev">
     <owl:sameAs rdf:resource="#CapitalOfUkraine"/>
</rdf:Description>.
```

2. Объявление тождественности классов **Employee** и **Person** (только в OWL Full):

```
<owl:Class rdf:ID="Employee">
     <owl:sameAs rdf:resource=
     "http://www.officeComp.com/office#Person"/>
</owl:Class>.
```

Свойство **owl:differentFrom** указывает, что два представителя не тождественны друг другу, т.е. ссылки URI в них указывают на разные ресурсы.

Значением свойств **rdfs:range** и **rdfs:domain** для свойства **owl:differentFrom** является **rdf:Thing**.

Пример 1.23. Использование свойства owl:differentFrom:

Задания представителя класса с именем **studIvanov1**, отличного от представителей **studIvanov2** и **studIvanov3**:

```
     <!-- Задание представителя класса Student
     с именем studIvanov1 -->
<inst:Student rdf:ID="studIvanov1">
          <!-- Задание полного имени -->
     <inst:fullName>
          <inst:firstName rdf:datatype="&xsd;token">
          Иван</inst:firstName>
          <inst:surName rdf:datatype="&xsd;token">
          Иванович</inst:surName>
          <inst:secondName rdf:datatype="&xsd;token">
          Иванов</inst:secondName>
     </inst:fullName>
          <!-- Задание даты рождения -->
     <inst:birthDate rdf:datatype="&xsd;date">
     1991-05-17</birthDate>
          <!-- Представитель studIvanov1 отличается
          от представителей studIvanov2 и studIvanov3 -->
     <owl:differentFrom rdf:resource="#studIvanov2"/>
     <owl:differentFrom rdf:resource="#studIvanov3"/>
</inst:Student>.
```

В тех онтологиях, где предполагается уникальность имен, использование свойства **owl:differentFrom** может привести к большому количеству предложениях, задающих отличие одного представления от другого. Для таких случаев в OWL введен класс **owl:AllDifferent**, для которого определено свойство **owl:distinctMembers**, связывающее экземпляр **owl:AllDifferent** со списком представителей (это свойство может использоваться только как субъект **owl:AllDifferent**). В этом списке указываются все представители, отличающиеся друг от друга.

Значениями свойств **rdfs:range** и **rdfs:domain** для свойства **owl:distinctMembers** являются соответственно **rdf:List** и **owl:AllDifferent**.

Пример 1.24. Использование класса owl:AllDifferent и свойства owl:distinctMembers:

Зададим всех отличных друг от друга представителей класса **Student**, указанных в примере 1.47, в одном списке в элементе **owl:AllDifferent**:

```
<owl:AllDifferent>
     <owl:distinctMembers rdf:parseType="Collection">
          <inst:Student rdf:about="#studIvanov1"/>
          <inst:Student rdf:about="#studIvanov2"/>
          <inst:Student rdf:about="#studIvanov3"/>
      </owl:distinctMembers>
</owl:AllDifferent>.
```

1.6. Перечислимые данные в OWL

В OWL можно использовать все типы данных, определенные в RDF. Кроме этого, в OWL DL и OWL Full можно использовать перечислимые данные, т.е. данные, задающиеся списками своих значений. Эти данные задаются в OWL с помощью свойства **owl:oneOf** (см. 1.3.1.2) и класса **rdf:List**.

Пример 1.25. Использование перечислимых данных в OWL DL и OWL Full:

Задание свойства типизированного данного (экзаменационные оценки), содержащего список из трех оценок: **4**, **4** и **5**:

```
<owl:DatatypeProperty rdf:ID="examMarks">
  <rdfs:range>
    <owl:DataRange>
          <!-- Задание списка оценок -->
      <owl:oneOf>
            <!-- Задание первой оценки -->
        <rdf:List>
          <rdf:first rdf:datatype=
          "&xsd;positiveInteger">4</rdf:first>
          <rdf:rest>
            <rdf:List>
                  <!-- Задание второй оценки -->
              <rdf:first rdf:datatype=
              "&xsd;positiveInteger">4</rdf:first>
              <rdf:rest>
                    <!-- Задание последней (третьей)
                    оценки -->
                <rdf:List>
                  <rdf:first rdf:datatype=
                  "&xsd;positiveInteger">5</rdf:first>
                  <rdf:rest rdf:resource="&rdf;nil"/>
                </rdf:List>
              </rdf:rest>
            </rdf:List>
          </rdf:rest>
        </rdf:List>
      </owl:oneOf>
    </owl:DataRange>
```

```
  </rdfs:range>
</owl:DatatypeProperty>.
```

2. ЗАДАНИЯ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

2.1. Инсталяция Jena Fuseki

Инсталируйте сервер Jena Fuseki, используя информацию из [1].

2.2. Изучение базы знаний PeriodicTable.owl

Изучите структуру онтологии PeriodicTable.owl.

2.3. Формирование интеллектуальных запросов

Сформируйте следующие запросы:

1. Выберите элементы онтологии с порядковыми номерами с 50 по 100.
2. Выберите имена элементов онтологии с фрагментом “silvery” в теге color.
3. Выберите имена элементов онтологии в стандартном состоянии газ (gas).

Содержание отчета:
Код запроса; полученный результат.