СПбГЭТУ «ЛЭТИ»
Кафедра Вычислительной техники

Магистерская программа
«Семантические технологии и многоагентные системы»

Дисциплина «Семантический Web»

Лекция 2

RDFS и OWL

RDFS

RDF Schema

RDF Схема (RDFS)

- **RDF** – формализм для описания ресурсов с помощью метаданных и способ его записи (XML, N-Triple, Turtle).
- Например:

  <geo:Volga> <rdf:type> <http://www.geodesy.org/river#River>.

- **Ограничения**: элементы словаря (subClassOf, type…) интерпретируются как произвольные бинарные отношения, им НЕ приписывается *явно формальная семантика*

- **RDF Схема** (RDFS) позволяет **определить базовый словарь терминов** и **отношений** между ними
  - придает конкретным предикатам и ресурсам RDF “дополнительный смысл” (семантику), которая специфицирует, как должен интерпретироваться термин

Словарь RDFS – классы и свойства

- Термины RDF-схемы:
- Классы:
  - **rdfs:Resource** - класс-ресурс включает **ВСЁ** (**ресурсом является ВСЕ!**)
  - **rdfs:Class** - класс Классов
  - **rdf:Property** - класс RDF-свойств (из пространства имен rdf: !)
  - **rdfs:Literal** - класс литеральных значений (текстовых строк или чисел)
  - **rdfs:Datatype** - класс типов данных RDF
- Свойства (отношения):
  - **rdf:type** - субъект является экземпляром класса (из rdf: !)
  - **rdfs:subClassOf** - субъект является подклассом класса
  - **rdfs:subPropertyOf** - субъект является подсвойством свойства
  - **rdfs:domain** - область определения свойства
  - **rdfs:range** - область значений свойства

RDFS - примеры определения сущностей

- *Термины RDF-схемы используются как конструкторы для создания словарей для различных предметных областей:*

<Person rdf:type rdfs:Class> // Person является классом

<hasColleague rdf:type rdf:Property> // hasColleague является свойством

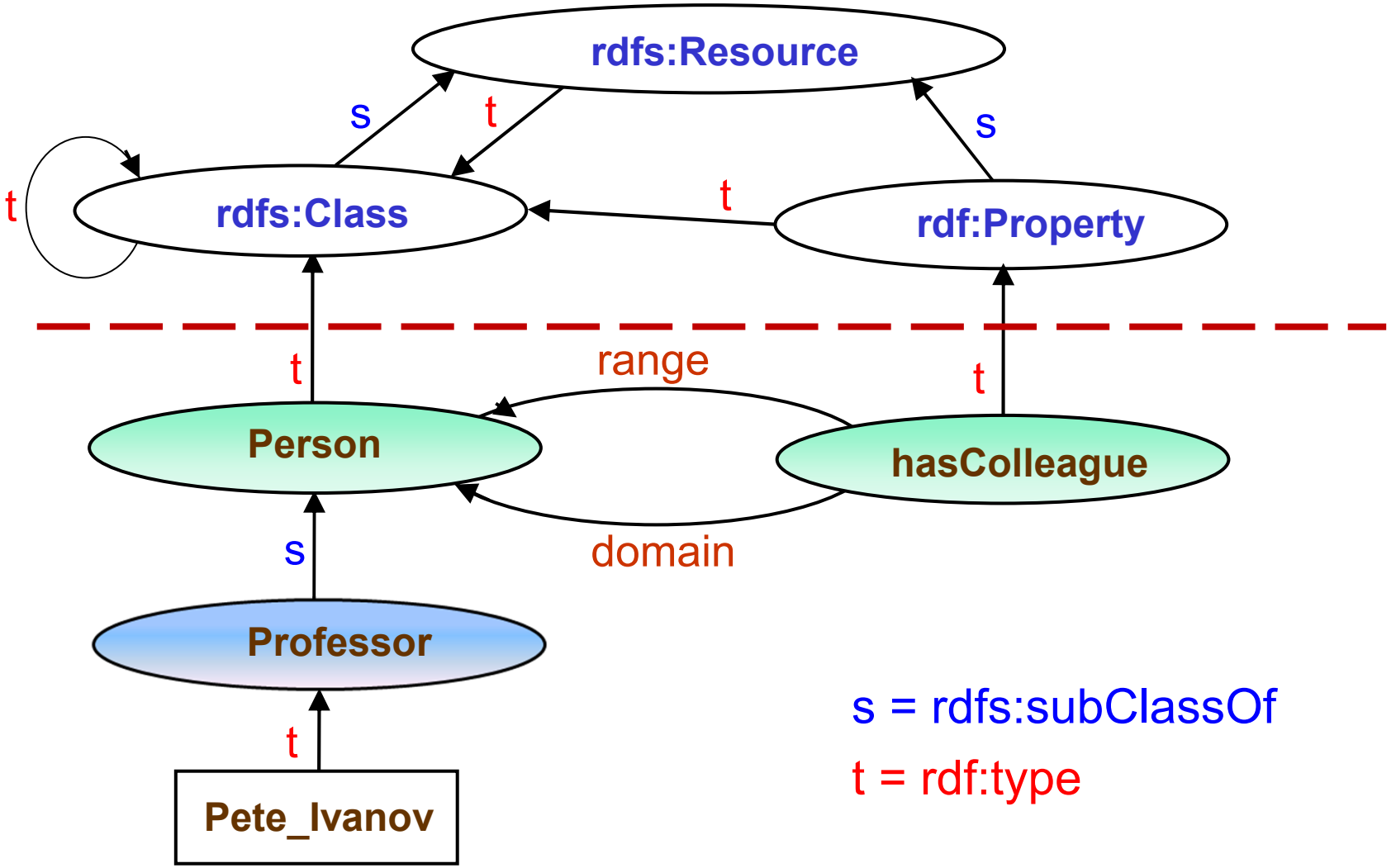
<Professor rdfs:subClassOf Person> // *Professor* является подклассом // класса *Person*

<Pete_Ivanov rdf:type Professor> // *Pete_Ivanov* является элементом // класса *Professor*

<hasColleague rdfs:domain Person> // свойством *hasColleague* могут // обладать элементы класса Person

<hasColleague rdfs:range Person> // значением свойства *hasColleague* // являются элементы класса *Person*

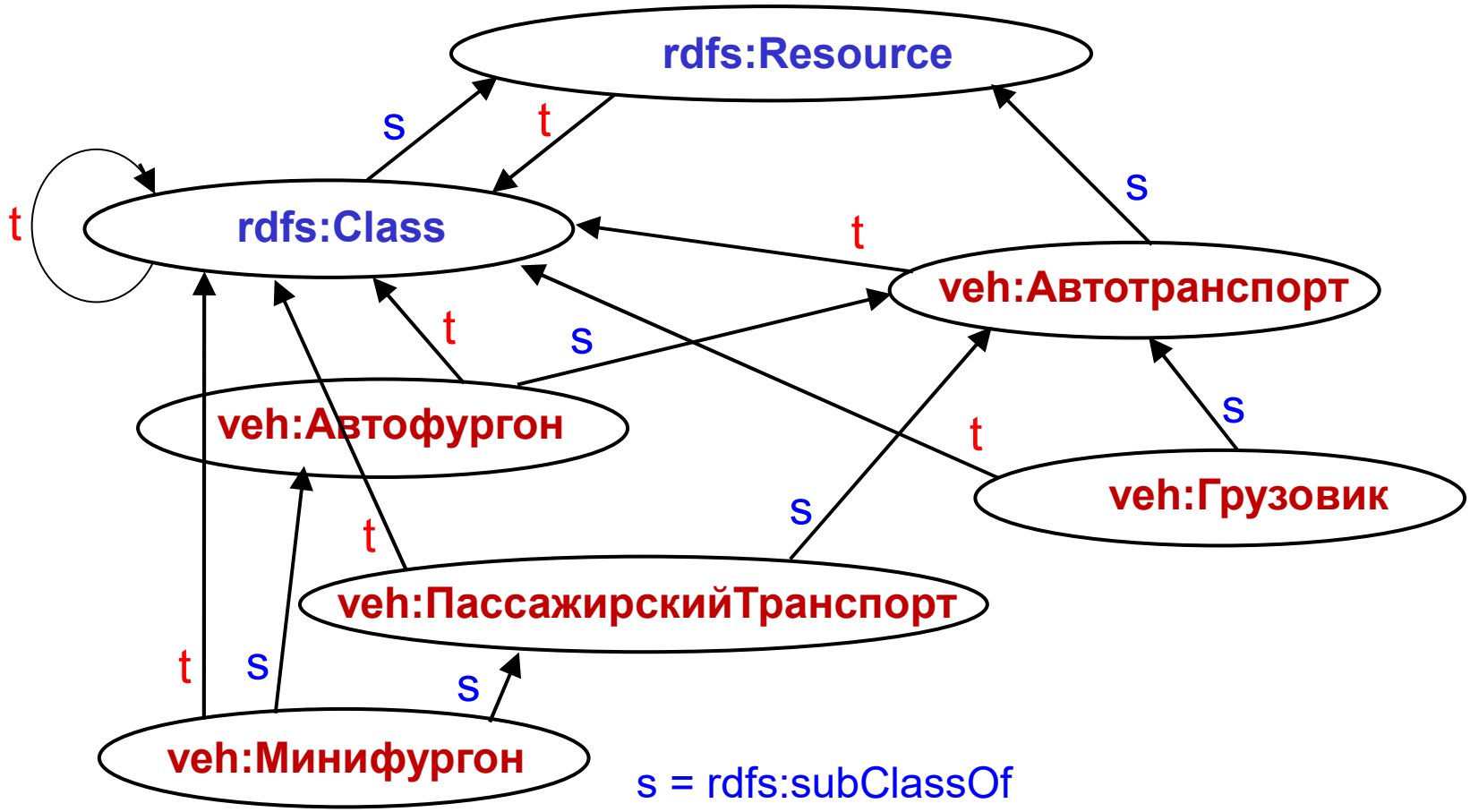
RDFS: соотношение сущностей



s = rdfs:subClassOf

t = rdf:type

RDFS: пример 2



s = rdfs:subClassOf

t = rdf:type

veh: - пространство имен, относящееся к транспортным средствам

Особенности RDF(S)

- Нет разницы между классами и экземплярами (индивидуами):

```
<Species,type,Class>  /*Биологический вид есть класс*/
<Lion,type,Species>   /*Лев – биологический вид*/
<Chandor,type,Lion>   /*Чандор – лев*/
```

- Свойства сами могут иметь свойства:

```
<hasDaughter,subPropertyOf,hasChild>
<hasDaughter,type,familyProperty>
```

- Нет различия между конструкторами языка и онтологическим словарем, поэтому конструкторы могут применяться к самим себе и друг к другу:

```
<type,range,Class>
<Property,type,Class>
<type,subPropertyOf,subClassOf>
```

Недостатки RDFS

- RDFS слишком беден, чтобы достаточно детально описывать ресурсы:
  - Отсутствуют ограничения, локализующие область определения и область значений *в зависимости от каких-то факторов*:
    - Нельзя сказать, что когда свойство **`hasChild`** применяется к людям, областью его значений является человек, а когда оно применяется к слонам - слон;
  - Отсутствуют ограничения существования/кардинальности
    - Нельзя сказать, что все экземпляры класса людей имеют мать, которая также относится к этому классу или что человек имеет в точности двух родителей
  - Отсутствуют транзитивные, инверсные или симметричные свойства
    - Нельзя сказать, что **`isPartOf`** – транзитивное свойство, **`hasPart`** – инверсное свойство с **`isPartOf`**, **`touchWith`** – симметричное свойство
- Трудно обеспечить поддержку рассуждений:
  - Для нестандартной семантики отсутствуют машины логического вывода

Язык онтологий для Семантического Web

OWL

Онтологии в информатике

"*Онтология – явная спецификация концептуализации*"

[T.Gruber, 1993]

- Онтология является инженерным (искусственно созданным) артефактом:
  - Состоит из конкретного словаря, используемого для описания определенной реальности и
  - Множества явных допущений, относящихся к подразумеваемому значению словаря
- Онтология описывает формальную спецификацию определенной предметной области:
  - Общее (разделяемое) понимание рассматриваемой предметной области
  - Формальная (допускающая машинную обработку) модель рассматриваемой предметной области

Типы знаний в онтологии

Онтологии обычно имеют два компонента:

- Имена понятий, важных для рассматриваемой предметной области:
  - **Слон –** класс (понятие), член которого является видом животных
  - **Травоядные –** понятие, членами которого являются в точности те животные, которые едят только растения или части растений
  - **Взрослый_Слон –** понятие, членами которого являются в точности те слоны, чей возраст больше 20 лет
- Базовые знания и ограничения предметной области:
  - **Взрослые_Слоны имеет вес не менее 2 000 кг**
  - **Все Слоны являются либо Африканскими_Слонами либо Индийскими_Слонами**
  - **Никакой индивидуум не может быть одновременно Травоядным и Плотоядным**

Языки представления (онтологических) знаний

- Разработано много языков явной спецификации концептуальных знаний:
  - **Основанные на графических нотациях:**
    - Семантические сети;
    - Карты понятий (Topic Maps) (http://www.topicmaps.org/);
    - UML;
    - RDF;
  - **Основанные на логике:**
    - Дескриптивные логики (OIL, DAML+OIL, OWL);
    - Правила (RuleML, LP/Prolog);
    - Логика первого порядка (KIF);
    - Концептуальные графы;
    - Логики высших порядков (LBase);
    - Неклассические логики (FLogic, немонотонные логики, модальности);
  - **Вероятностные/нечеткие**
- Степень формальности языков варьируется в широких пределах
  - Повышение уровня формальности делает языки в большей степени пригодными для машинной обработки (автоматических рассуждений)

Многие языки используют *объектно-ориентированную парадигму* модели мира

- Объекты/Экземпляры/Индивидуумы
  - элементы области дискурса
  - в логике первого порядка - эквивалентны константам
- Типы/Классы/Понятия
  - множества объектов, имеющих общие характеристики
  - в логике первого порядка - эквивалентны унарным предикатам
- Отношения/Свойства/Роли
  - множества пар (троек) объектов
  - в логике первого порядка - эквивалентны бинарным предикатам
- Такие языки:
  - хорошо понимаемы;
  - формально специфицированы;
  - (относительно) легки в использовании
  - поддаются машинной обработке

Требования к языку онтологий для Web

Язык Web-онтологий должен:

- Расширять существующие стандарты Web
  - такие как XML, RDF, RDFS
- Быть простым для понимания и использования
  - должен базироваться на хорошо известных способах представления знаний
- Быть **формально специфицированным**
- Обладать “адекватной” выразительной мощностью
- Обеспечить поддержку автоматических рассуждений

OWL: предыстория

- Чтобы удовлетворить указанным требованиям на первом этапе были разработаны два языка:
  - OIL - разработан группой (в основном) Европейских исследователей
  - DAML-ONT - разработан группой (в основном) американских исследователей (в рамках выполняемой DARPA программы DAML)
- Усилия были объединены для разработки DAML+OIL
  - Разработка была выполнена “Объединенным EU/US Комитетом по агентным языкам разметки - Agent Markup Languages”
  - Расширяет (DL подмножество) RDF
- DAML+OIL был представлен W3C как основа для стандартизации
  - была сформирована рабочая группа Web-Ontology (WebOnt)
  - WebOnt на основе DAML+OIL разработала язык OWL
  - **OWL - W3C Recommendation !!!**

Язык OWL

- Три уровня (диалекта) OWL
  - OWL Lite – «*простое для реализации*» подмножество OWL DL
  - OWL DL - ограничен фрагментом логики первого порядка (¼ DAML+OIL)
  - OWL full - объединение OWL-синтаксиса и RDF
- Уровни семантики:
  - OWL DL составляет ¼ OWL full
  - семантика DL четко завиксирована
- OWL DL основан на **Дескриптивной Логике *SHIQ***
  - В действительности эквивалентен $SHOIN(D_n)$ DL
- Достоинства OWL DL:
  - строго определенная семантика;
  - хорошо осмыслены формальные свойства (сложность, разрешимость);
  - известны алгоритмы рассуждений;
  - имеются хорошие реализации систем на основе OWL DL;

W3C спецификации OWL 2 Web Ontology Language (11.12.2012)

- OWL 2 Web Ontology Language **Document Overview** (Second Edition) (https://www.w3.org/TR/owl2-overview/);
- OWL 2 Web Ontology Language **XML Serialization** (Second Edition) (https://www.w3.org/TR/owl2-xml-serialization/)
- OWL 2 Web Ontology Language **Manchester Syntax** (Second Edition) (https://www.w3.org/TR/owl2-manchester-syntax/)
- OWL 2 Web Ontology Language **Structural Specification and Functional-Style Syntax** (Second Edition) (https://www.w3.org/TR/owl2-syntax/)
- OWL 2 Web Ontology Language **Conformance** (Second Edition) (https://www.w3.org/TR/owl2-conformance/)
- OWL 2 Web Ontology Language **Mapping to RDF Graphs** (Second Edition) (https://www.w3.org/TR/owl-mapping-to-rdf/)
- OWL 2 Web Ontology Language **Primer** (Second Edition) (https://www.w3.org/TR/owl2-primer/)
- OWL 2 Web Ontology Language **Direct Semantics** (Second Edition) (https://www.w3.org/TR/owl2-direct-semantics/)
- OWL 2 Web Ontology Language **RDF-Based Semantics** (Second Edition) (https://www.w3.org/TR/owl2-rdf-based-semantics/)
- OWL 2 Web Ontology Language **Profiles** (Second Edition) (https://www.w3.org/TR/owl2-profiles/)

OWL: уровни (диалекты)

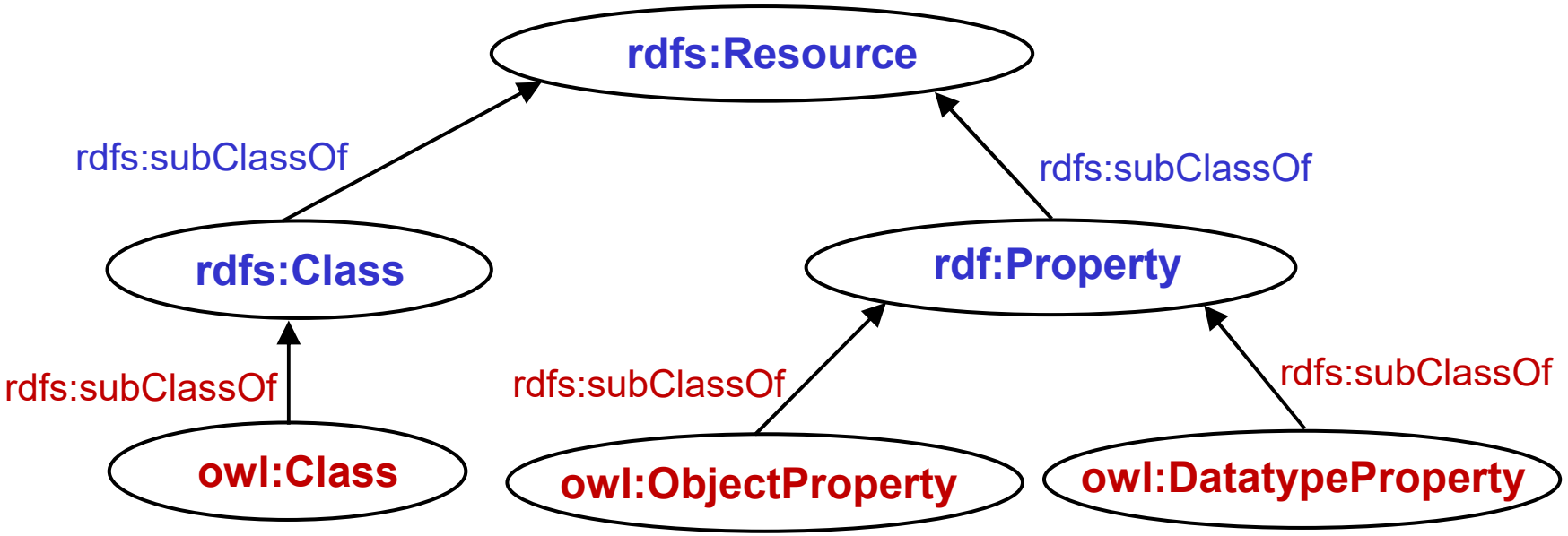
В **OWL 1.0** выделено ***три диалекта*** с разными выразительными возможностями:

- **OWL Lite:**
  - – Классификационные иерархии;
  - – Простые ограничения;
- **OWL DL:**
  - – Максимальная выразительность при *сохранении разрешимости*;
  - – Стандартная формализация;
- **OWL Full:**
  - – Очень высокая выразительность (метаклассы, классы как значения);
  - – Полная синтаксическая свобода RDF (самомодифицируемость);
  - – ***Потеря разрешимости!!!*** Алгоритмы логического вывода *общем случае становятся неэффективными!*

В языке **OWL 2.0** (OWL 2), принятом **11.12.2012**, выделено 3 профиля:

- **OWL 2 EL** – для приложений, использующих онтологии *с очень большим числом свойств и/или классов*. Выразительные возможности допускают решение основных задач логического вывода за полиномиальное от размера онтологии время.
- **OWL 2 QL** – для приложений, работающих с *очень большими объемами данных*, в которых *наиболее важной задачей вывода является ответ на запросы*. Выразительная мощность профиля достаточно ограничена.
- **OWL 2 RL** – для приложений, требующих *масштабируемых рассуждений без значительной потери выразительной мощности*. Модули рассуждений OWL 2 могут быть реализованы с использованием машин вывода, основанных на правилах.

Соотношение элементов OWL и RDF(S)



OWL-онтология и OWL-документ. Варианты синтаксиса

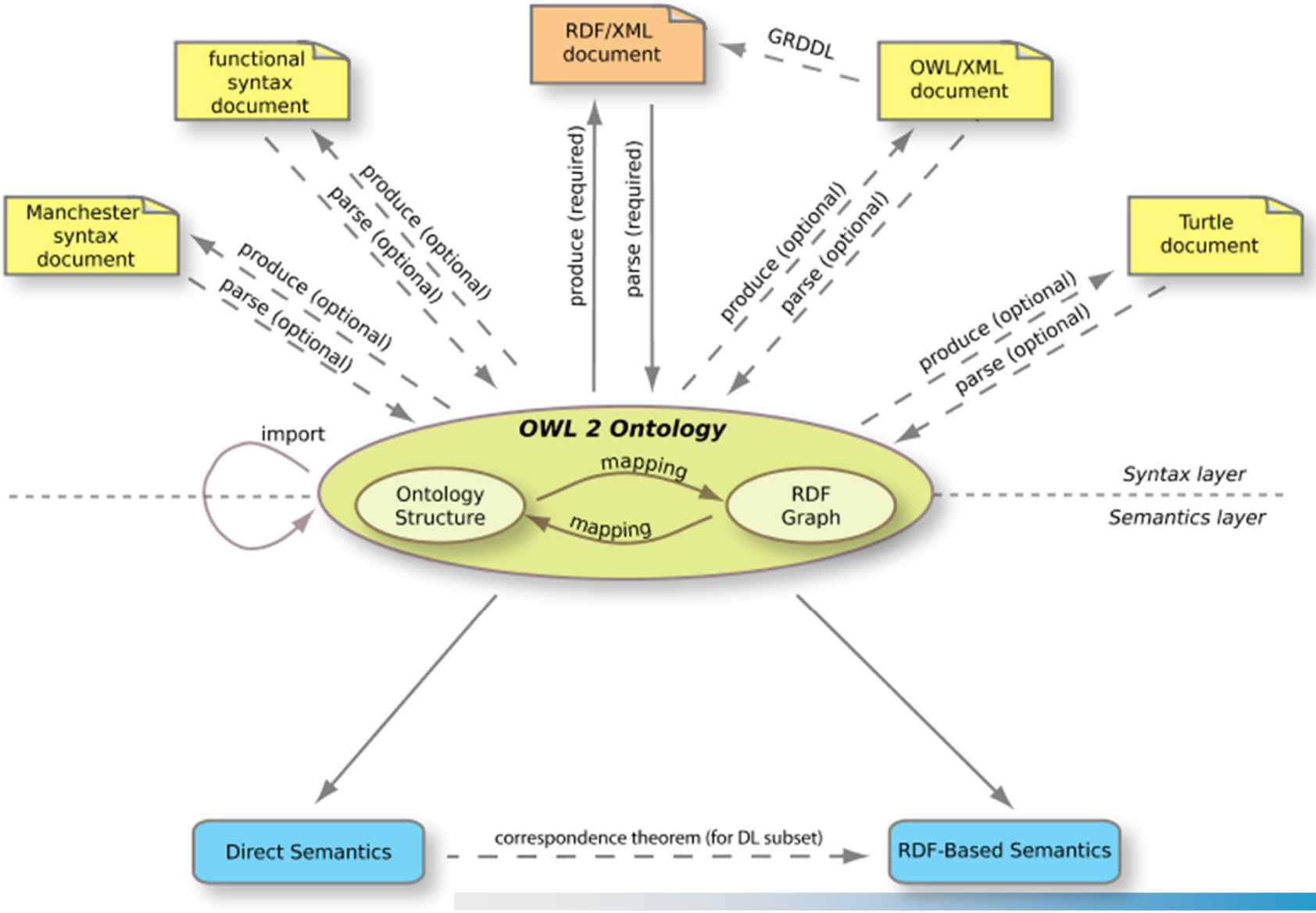
- ***OWL онтология*** – абстрактное понятие, определенное на концептуальном уровне в структурной спецификации;
- Каждая онтология связана с ***документом онтологии***, который *физически содержит* онтологию (хранит ее *определенным образом*)
  - термин "***документ онтологии***" исходит из представления о хранении онтологии в физическом текстовом документе с использованием того или иного синтаксиса OWL 2;
  - при этом *инструменты OWL 2* могут разрабатывать *другие типы документов онтологий*, вводя другие способы физического хранения онтологий;

Варианты синтаксиса OWL документов

Синтаксис	Цель	Спецификация
RDF/XML	Обмен. Может быть записан и прочитан всеми программами, совместимыми с OWL 2	Mapping to RDF Graphs, RDF/XML
OWL/XML	Облегчить обработку с использованием XML инструментов	XML Serialization
Функциональный Functional Syntax	Улучшить наглядность формальной структуры онтологий	Structural Specification
Манчестерский Manchester Syntax	Улучшить читабельность DL онтологий	Manchester Syntax
Turtle	Улучшить читабельность RDF троек	Mapping to RDF Graphs, Turtle

Связь OWL онтологии и OWL документов

(https://www.w3.org/TR/owl2-overview/)



OWL-документ с синтаксисом RDF/XML: Пространства имен

- **OWL-документы** с синтаксисом RDF/XML являются **XML-документами** с корневым элементом rdf:RDF

- В этом элементе обычно специфицируются используемые пространства имен. Основные из них:

```
<rdf:RDF
   xmlns:owl ="http://www.w3.org/2002/07/owl#"
   xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

OWL-документ: преамбула

- OWL-онтология может начинаться с множества служебных утверждений
- Эти утверждения группируются в элементе **`owl:Ontology`**, содержащем *комментарии*, *управление версиями* и *включение других онтологий*
- Например:

```
<owl:Ontology rdf:about="">
    <rdfs:comment>An example OWL ontology</rdfs:comment>
    <owl:priorVersion rdf:resource="http://www.mydomain.org/uni-ns-old"/>
    <owl:imports rdf:resource="http://www.mydomain.org/persons"/>
    <rdfs:label>University Ontology</rdfs:label>
</owl:Ontology>
```

- **owl:imports** – Перечисляет другие онтологии, содержание которых подразумевается как часть текущей онтологии. Единственное утверждение, существенное **для логической обработки онтологии**.

OWL: Элемент **owl:Class**

- Классы определяются, используя элемент **owl:Class**, который является подклассом **rdfs:Class**
- Пример:

```
<owl:Class  rdf:ID="associateProfessor">
   <rdfs:subClassOf  rdf:resource="#academicStaffMember"/>
</owl:Class>
```

- Атрибут **rdf:ID**
  - вводит название класса
  - название позволяет ссылаться на этот класс внутри данного документа так:
    - **rdf:resource="#associateProfessor"**
  - другие (внешние) онтологии могут ссылаться на это название, используя полную ссылку:
    - **rdf:resource="http://www.mydomain.org/uni-ns#associateProfessor"**

OWL: Элемент owl:disjointWith

- Элемент **owl:disjointWith** позволяет сказать, что данный класс не пересекается (не имеет общих экземпляров) с другими
- Например, класс **associateProfessor** не пересекается с классами **professor** и **assistantProfessor**
  - эти элементы могут включаться в приведенное выше определение или добавляться путем ссылки на **id**, используя **rdf:about**
  - механизм наследуется от RDF:

```
<owl:Class   rdf:about="associateProfessor">
   <owl:disjointWith  rdf:resource="#professor"/>
   <owl:disjointWith  rdf:resource="#assistantProfessor"/>
</owl:Class>
```

OWL: эквивалентность классов. Классы owl:Thing и owl:Nothing

- Эквивалентность классов может быть определена, используя элемент owl:equivalentClass:

```
<owl:Class rdf:ID="faculty">
  <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```

- Имеется два заранее определенных класса, аналогичных универсуму и пустому множеству в теории множеств:
  - owl:Thing – наиболее общий класс, содержащий **все** (все является вещью);
  - owl:Nothing – пустой класс

- Каждый класс является подклассом **owl:Thing** и суперклассом **owl:Nothing**

OWL: Описание свойств

- В OWL выделено **два вида свойств**:
  - Свойства типа данных (Datatype properties) – связывают объекты со значениями базовых типов данных
    - например: phone, title, age
  - Объектные свойства (Object properties) – связывают объекты с другими объектами
    - например: **isTaughtBy**, **supervises**
- OWL
  - не имеет предопределенных типов данных и собственных способов их определения;
  - использует типы данных XML Schema;
  - определяемые пользователем типы данных можно собрать в XML-схему, а затем использоваться в OWL-онтологии
- Пример свойства типа данных:

```
<owl:DatatypeProperty rdf:ID="age">
  < rdfs:range rdf:resource="http://www.w3.org/2001/XLMSchema#nonNegativeInteger"/ >
</owl:DatatypeProperty >
```

Типы данных XML Schema, зарезервированные в словаре OWL 2

xsd:anyURI

xsd:base64Binary

xsd:boolean

xsd:byte

xsd:dateTime

xsd:decimal

xsd:double

xsd:float

xsd:dateTimeStamp

xsd:integer

xsd:language

xsd:length

xsd:long

xsd:maxInclusive

xsd:maxLength

xsd:minExclusive

xsd:hexBinary

xsd:Name

xsd:NCName

xsd:negativeInteger

xsd:int

xsd:nonPositiveInteger

xsd:NMTOKEN

xsd:nonNegativeInteger

xsd:maxExclusive

xsd:minInclusive

xsd:minLength

xsd:normalizedString

xsd:pattern

xsd:positiveInteger

xsd:short

xsd:string

xsd:token

xsd:unsignedByte

xsd:unsignedInt

xsd:unsignedLong

xsd:unsignedShort

OWL: Пример описания объектного свойства

- Объектное свойство **isTaughtBy** (преподаваться)
- Пример объектного свойства:

```
<owl:ObjectProperty      rdf:ID="isTaughtBy">
  <rdfs:subPropertyOf    rdf:resource="#involves"/>
  <owl:domain   rdf:resource="#course"/>
  <owl:range    rdf:resource="#academicStaffMember"/>
</owl:ObjectProperty>
```

Характеристики свойств

- Поскольку свойства выражают отношения между сущностями, они сами могут обладать известными в математике свойствами:
  - Транзитивность (TransitiveProperty);
  - Симметричность (SymmetricProperty);
  - Функциональность (FunctionalProperty);
  - Инверсность (inverseOf);
  - Инверсная функциональность (InverseFunctionalProperty)

  - Эти свойства очень важны для поддержки автоматических рассуждений (ризонинга)

Характеристики свойств: Транзитивность

- Свойство P транзитивно (TransitiveProperty), если для любых x, y и z: справедливо:
  - P(x, y) & P(y, z) – > P(x, z)

- Пример:

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

Характеристики свойств: Симметричность

- Свойство P симметрично (SymmetricProperty), если для любых x и y справедливо:

  P(x, y) <– > P(y, x,)

- Пример:
  - Отношение **bordersWith** («граничит с»)

```
<owl:ObjectProperty rdf:ID="bordersWith">
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
  <rdfs:domain rdf:resource=". . ."/>
  <rdfs:range rdf:resource=". . ."/>
  </owl:ObjectProperty>
```

Характеристики свойств: Функциональность

- Свойство P функционально (FunctionalProperty), если для любых x, y и z справедливо:

  P(x, y) & P(x, z) – > y = z

- Пример:
  - Свойство **yearOfBirth** («год рождения»)

```
<owl:DatatypeProperty rdf:ID=" yearOfBirth">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource=". . ."/>
  <rdfs:range rdf:resource=". . ."/>
  </owl:DatatypeProperty>
```

OWL: Инверсные свойства

- Элемент **inverseOf** позволяет определять свойство как инверсное другому свойству;
  - например: **isTaughtBy** (преподается) и **teaches** (преподает)

```
<owl:ObjectProperty rdf:ID="teaches">
  <rdfs:domain rdf:resource="#academicStaffMember"/>
  <rdfs:range  rdf:resource="#course"/>
  <owl:inverseOf   rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```

- Области определения и область значений могут наследоваться от инверсного свойства (перестановкой области определения и области значений)

Характеристики свойств: Инверсная функциональность

- Свойство P инверсно функционально (InverseFunctionalProperty), если для любых x, y и z справедливо:

  P(y, x) и P(z, x) – > y = z

```
<owl:ObjectProperty rdf:ID=«hasManufacturer" />
<rdfs:domain rdf:resource="#car"/>
    <rdfs:range rdf:resource="#manufacturer"/>
 .   .   .
<owl:ObjectProperty rdf:ID="makesCar">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
    <owl:inverseOf rdf:resource="#hasManufacturer" />
</owl:ObjectProperty>
```

OWL: Эквивалентные свойства

- Эквивалентность свойств может быть определена используя элемент **owl:equivalentProperty**:

```
<owl:ObjectProperty  rdf:ID="lecturesIn">
   <owl:equivalentProperty  rdf:resource="#teaches"/>
</owl:ObjectProperty>
```

OWL: Ограничения свойств

- Классы можно определять путем накладываемых на свойство ограничений. Для этого используется элемент **owl:Restriction**
- Элемент **owl:Restriction** в общем случае содержит элемент **owl:onProperty** и одно или более объявлений ограничений
- Два типа ограничений:
  - ограничения на тип значений свойства
    - **owl:allValuesFrom**;
    - **owl:hasValue**;
    - **owl:someValuesFrom**;
  - ограничения кардинальности – числа значений

OWL: Ограничения свойств - **allValuesFrom**

- **rdfs:subClassOf** позволяет специфицировать класс C как подкласс другого класса C'. Объявление класса C, элементы которого удовлетворяют определенным условиям, эквивалентно утверждению, что C является подклассом C', в котором собраны все объекты, удовлетворяющие этим условиям.

- Пример: курсы первого года обучения, читаемые только профессорами:

```
<owl:Class  rdf:about="#firstYearCourse">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty     rdf:resource="#isTaughtBy"/>
      <owl:allValuesFrom  rdf:resource="#Professor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

- **owl:allValuesFrom** описывает класс возможных значений, которые может принимать свойство, специфицированное элементом **owl:onProperty** (значением свойства **isTaughtBy** могут быть только профессора).

OWL: Ограничения свойств - **hasValue**

- Объявление, что курсы по математике читаются профессором, имеющим идентификатор 949352:

```
<owl:Class rdf:about="#mathCourse">
   <rdfs:subClassOf >
     <owl:Restriction >
        <owl:onProperty  rdf:resource="#isTaughtBy"/ >
        <owl:hasValue    rdf:resource="#949352"/ >
     </owl:Restriction >
   </rdfs:subClassOf >
</owl:Class >
```

- **owl:hasValue** утверждает, что свойство, специфицированное элементом **owl:onProperty** должно иметь конкретное значение

OWL: Ограничения свойств - **someValuesFrom**

- Пример:
  - «Преподаватели, читающие по крайней мере один курс уровня undergraduate»:

```
<owl:Class     rdf:about="#academicStaffMember">
   <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty       rdf:resource="#teaches"/>
        <owl:someValuesFrom rdf:resource="#undergraduateCourse"/>
      </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class >
```

OWL: Ограничения кардинальности свойств

- Пример «Каждый курс читается по крайней мере одним преподавателем»:

```
<owl:Class rdf:about="#course">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty   rdf:resource="#isTaughtBy"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
       1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

- Литерал «1» интерпретируется как неотрицательное целое (nonNegativeInteger), а не как строка или еще что-то;
- Используется объявление пространства имен **xsd**, сделанное в элементе-заголовке для ссылки на документ XML-Schema.

OWL: Ограничения кардинальности свойств

- «Кафедра должна включать не менее десяти и не более тридцати сотрудников»:

```
<owl:Class rdf:about="#department">
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#hasMember"/>
       <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger«>
   10
       </owl:minCardinality>
       <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
        30
       </owl:maxCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

OWL: Описания сложных классов

- OWL поддерживает конструирование классов с использованием теоретико-множественных операторов:
  - Объединение (**`unionOf`**);
  - Пересечение (**`intersectionOf`**);
  - Дополнение (**`complementOf`**)

Описания сложных классов: Объединение

- Пример:
  - Класс «Фрукты» является объединением классов «Сладкие_фрукты» и «Несладкие_фрукты»

```
<owl:Class rdf:ID="Fruit">
    <owl:unionOf   rdf:parseType="Collection">
       <owl:Class rdf:about="#SweetFruit" />
       <owl:Class rdf:about="#NonSweetFruit" />
     </owl:unionOf>
</owl:Class>
```

Описания сложных классов: Пересечение

- Пример:
  - Класс «Белое_вино» является пересечением класса «Вино» и класса, у которого свойство цвет имеет значение «Белое»

```
<owl:Class rdf:ID="WhiteWine">
 <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#Wine" />
   <owl:Restriction>
       <owl:onProperty rdf:resource="#hasColor" />
       <owl:hasValue rdf:resource="#White" />
   </owl:Restriction>
 </owl:intersectionOf>
</owl:Class>
```