KDI
RDF and OWL

Fausto Giunchiglia and Mattia Fumagallli

University of Trento

Index

1. XML
   1.1. From HTML to XML
   1.2. Similarities and differences
   1.3. XML features and limits
2. RDF
   2.1. Syntax
      2.1.1. Language and data model
      2.1.2. Reification
      2.1.3. Containers and collections
   2.2. Semantics
      2.2.1. RDF schema
   2.3. Reasoning
      2.3.1. Inference system
      2.3.2. Inference rules
   2.4. Exercises
3.OWL
3.1. The OWL Full Language
3.2. OWL DL and OWL lite
3.3. Exercises

XML

HTML vs. XML

The Adventures of Tom Sawyer

Front piece of *The Adventures of Tom Sawyer*

Author	Mark Twain
Cover artist	created by Mark Twain
Country	United States
Language	English, Limited Edition(Spanish)
Genre	Bildungsroman, picaresque, satire, folk, children's novel
Publisher	American Publishing Company
Publication date	1876[1]
OCLC	47052486
Dewey Decimal	Fic. 22
LC Class	PZ7.T88 Ad 2001
Followed by	*Adventures of Huckleberry Finn*
Text	*The Adventures of Tom Sawyer* at Wikisource

HTML: focus on presentation

```
<h2>The adventures of Tom Sawyer</h2>
...
<b>Author: </b> Mark Twain <br>
<b>Cover artist: </b> created by <a href="http://...">Mark Twain </a>
...
```

XML: focus on metadata

```
<book>
        <title> The adventures of Tom Sawyer </title>
        <author> Mark Twain </author>
        <genre> Bildungsroman </genre>
        <genre> picaresque </genre>
        ...
        <publisher> American Publishing Company </publisher>
        <year>1876</year>
</book>
```

HTML vs. XML: similarities and differences

Similarities

- They both use of tags
- Tags may be nested
- Human can read and interpret both HTML and XML quite easily
- Machines can read and interpret only to some extent

Differences

- HTML is to tell machines about how to interpret formatting for graphical presentation
- XML is to tell machines about metadata content and relationships between different pieces of information
- XML allows the definition of constraints on values
- HTML tags are fixed, while XML tags are user defined

More about XML

- XML meta markup language: language for defining markup languages
- Query languages for XML:
  - Xquery
  - XQL
  - XML-QLs
- Style sheets can be written in various languages to define how to present XML to humans:
  - CSS2 (cascading style sheets level 2)
  - XSL (extensible stylesheet language)
- Transformations: XSLT specifies rules to transform an XML document to:
- another XML document
- an HTML document
- plain text

XML in a nutshell

XML features:

- A metalanguage that allows users to define markup
- It separates content and structure from formatting
- It is the de facto standard for the representation and exchange of structured information on the Web
- It is supported by query languages

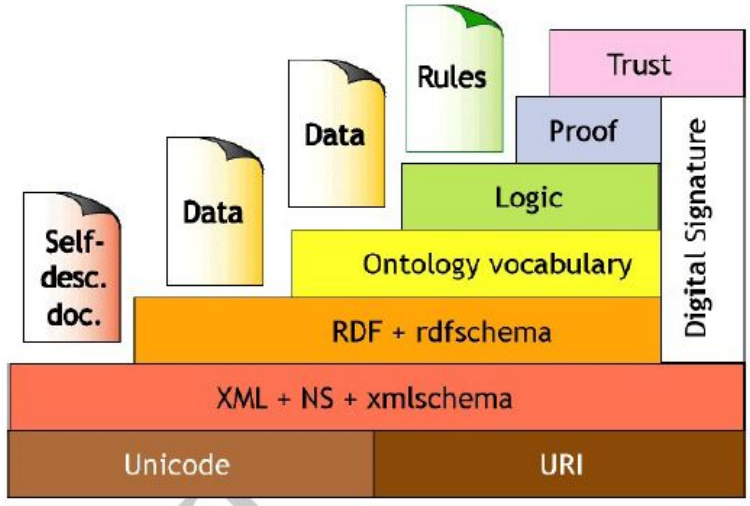
XML limits:

- The semantics of XML documents is not accessible to machines
- The nesting of tags does not have standard meaning
- Interoperability is possible if there is a shared understanding of the vocabulary

RDF

RDF and the Semantic Web

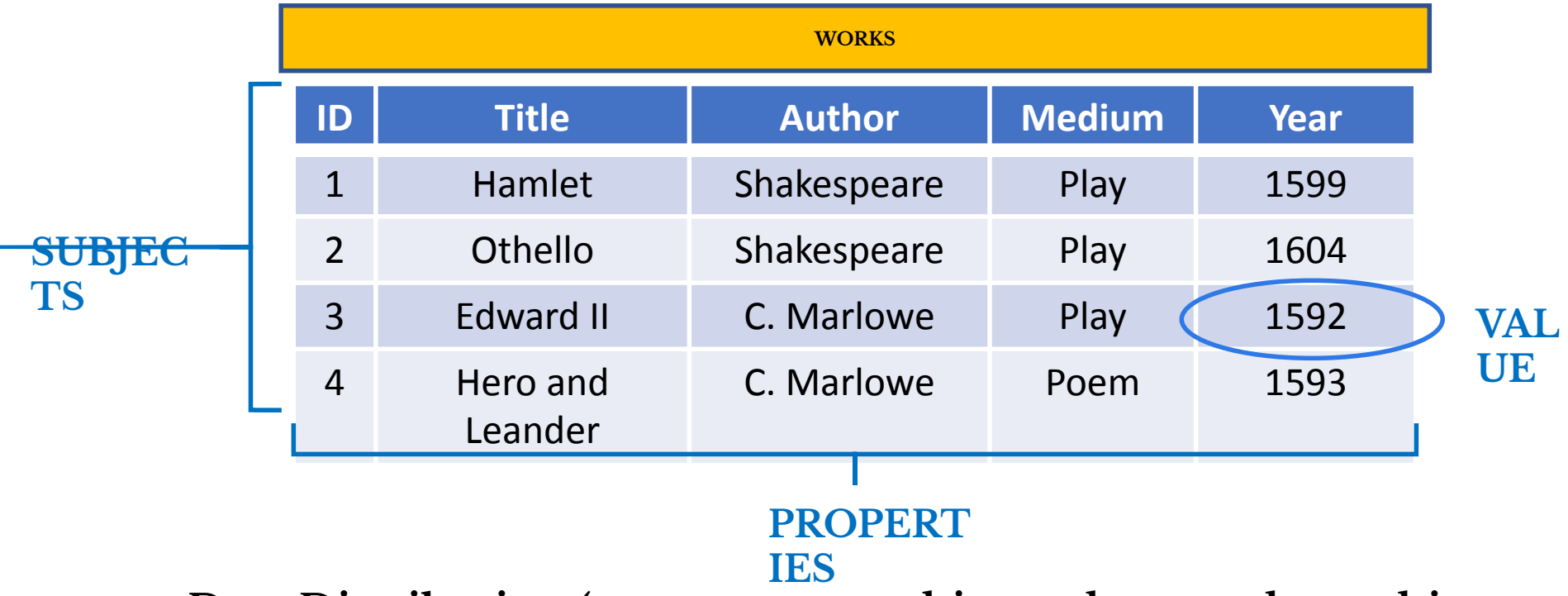
RDF (Resource Description Framework) is at the basis of the Semantic Web



Definitions

- A language for representing Web resources and information about them in the form of metadata [RDF Primer]
- A language to represent all kinds of things that can be identified on the Web [RDF Primer]
- A domain independent data model for representing information on the Web [G. Antoniou and F. van Harmelen, 2004]
- A language with an underlying model designed to publish data on

Distributing Data Across the Web



- **Data Distribution (over many machines where each machine maintains a part)**
- **row by row**
- **column by column**
- **cell by cell** (the strategy taken by RDF):
- a global identifier for the column headings
- a global identifier for the row headings
- a global identifier for non-literal values

RDF syntax

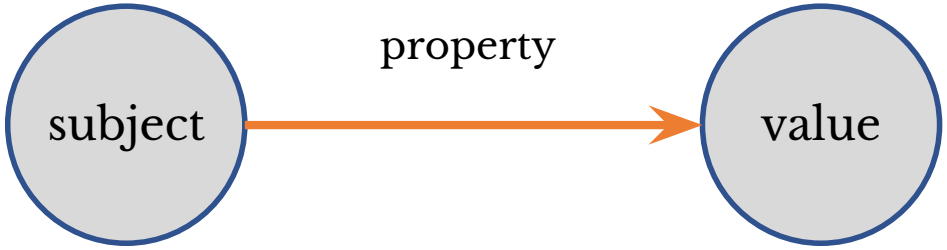
RDF language and data model

RDF language

- A language for representing data in the Semantic Web
- A language for expressing simple statements of the form subject-property-value (binary predicates)
- The capability to perform inference on statements

RDF data model

- The data model in RDF is a graph data model
- An edge with two connecting nodes forms a triple



RDF language

Formal syntax:

- RDF has been given a syntax in XML and inherits all its benefits
- Statements in RDF are machine comprehensible

Resources:

- An object, an entity or anything we want to talk about (e.g. authors, books, publishers, places, people, facilities)

Properties:

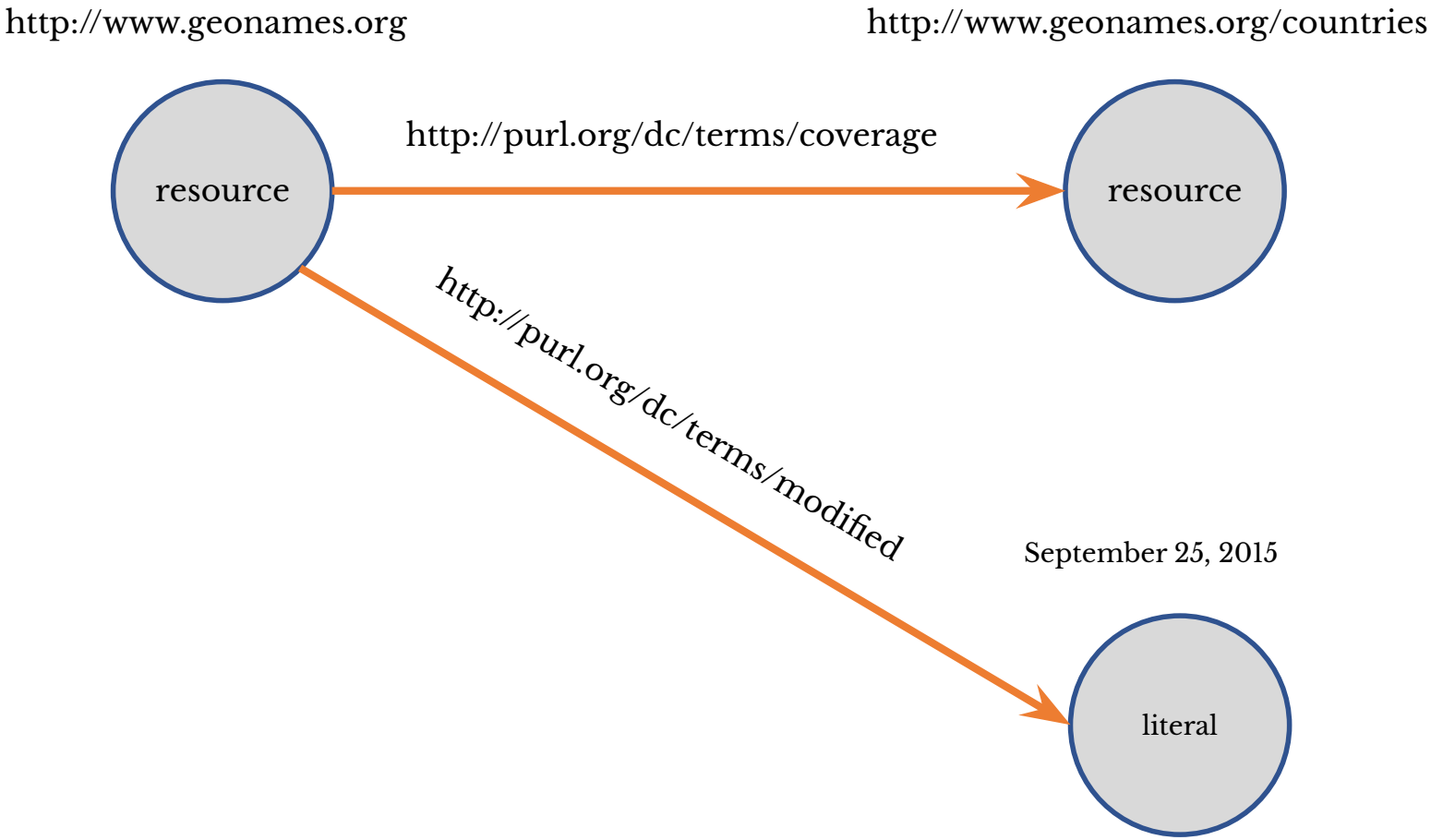
- They codify relations (e.g. written-by, friend-of, located-in, …) and attributes (e.g. age, date of birth, length …)

Statements:

- Statements assert the properties of resources in form of triples subject-property-value
- Every resource and property has a URI (an URL or any other identifier)
- Values can be resources (for relations) or literals (for attributes)

RDF data model

RDF as graph


http://www.geonames.org
http://www.geonames.org/countries
resource
http://purl.org/dc/terms/coverage
resource
http://purl.org/dc/terms/modified
September 25, 2015
literal

XML syntax example

```
<?xml version="1.0"?>

 <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:dc="http://purl.org/dc/terms#">

<rdf:Description rdf:about="http://www.geonames.org">
    <rdfs:label>GeoNames</rdfs:label>
    <dc:coverage
rdf:resource="http://www.geonames.org/countries"/>
    <dc:modified>September 25, 2015</dc:modified>
</rdf:Description>

</rdf:RDF>
```

RDF/XML elements

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:uni="http://www.mydomain.org/uni-ns">

    <rdf:Description  rdf:about="CIT1111">
        <uni:courseName>Discrete Matematics</uni:courseName>
        <uni:isTaughtBy rdf:resource="#949318">
        <uni:age rdf:datatype="&xsd:integer">27<uni:age>
    </rdf:Description>

    <rdf:Description  rdf:ID="#949318">
        <uni:name>David Billington</uni:name>
        <uni:title>Associate Professor</uni:title>
        <uni:age rdf:datatype="&xsd:integer">27<uni:age>
    </rdf:Description>
</rdf:RDF>
```

NAMESPACES

RESOURCE HAS BEEN DEFINED ELSEWHERE

RESOURCE IS DEFINED HERE

URI or fragment of it

VALUE

RELATION

ATTRIBUTE

DATA TYPE

RDF typing

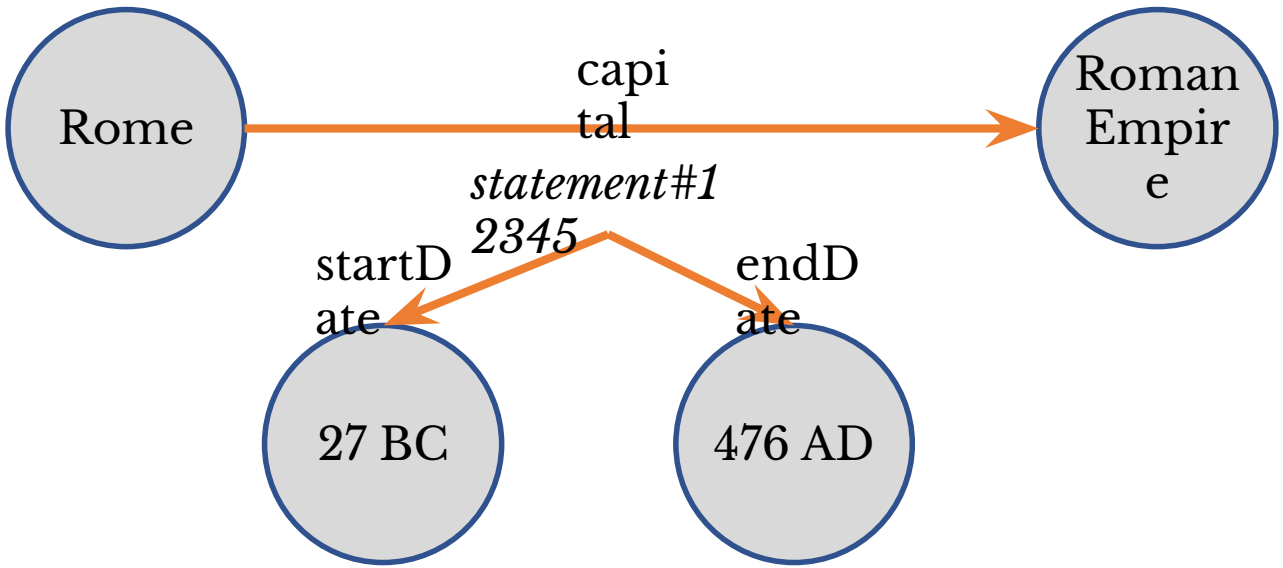
```
<rdf:Description rdf:ID="CIT1111">
        <rdf:type
    rdf:resource="http://www.mydomain.org/uni-ns#cours
    e"/>
        <uni:courseName>Discrete
    Maths</uni:courseName>
        <uni:isTaughtBy rdf:resource="#949318"/>
</rdf:Description>

<rdf:Description rdf:ID="949318">
        <rdf:type
    rdf:resource="http://www.mydomain.org/uni-ns#lectur
    er"/>
        <uni:name>David Billington</uni:name>
        <uni:title>Associate Professor</uni:title>
</rdf:Description>
```

RDF Reification

Reification can be used to represent:

- Generic statements about statements
- Structured attributes (e.g. address)
- Units of measure
- Provenance information
- Time validity and other contextual information



RDF semantics

RDF schema

RDF

- **RDF is a universal language** that lets users describe resources in their own vocabularies
- RDF by default does not assume, nor defines semantics of any particular application domain

RDF schema (RDFS)

A language defined to provide mechanisms to add semantics to RDF resources, in terms of:

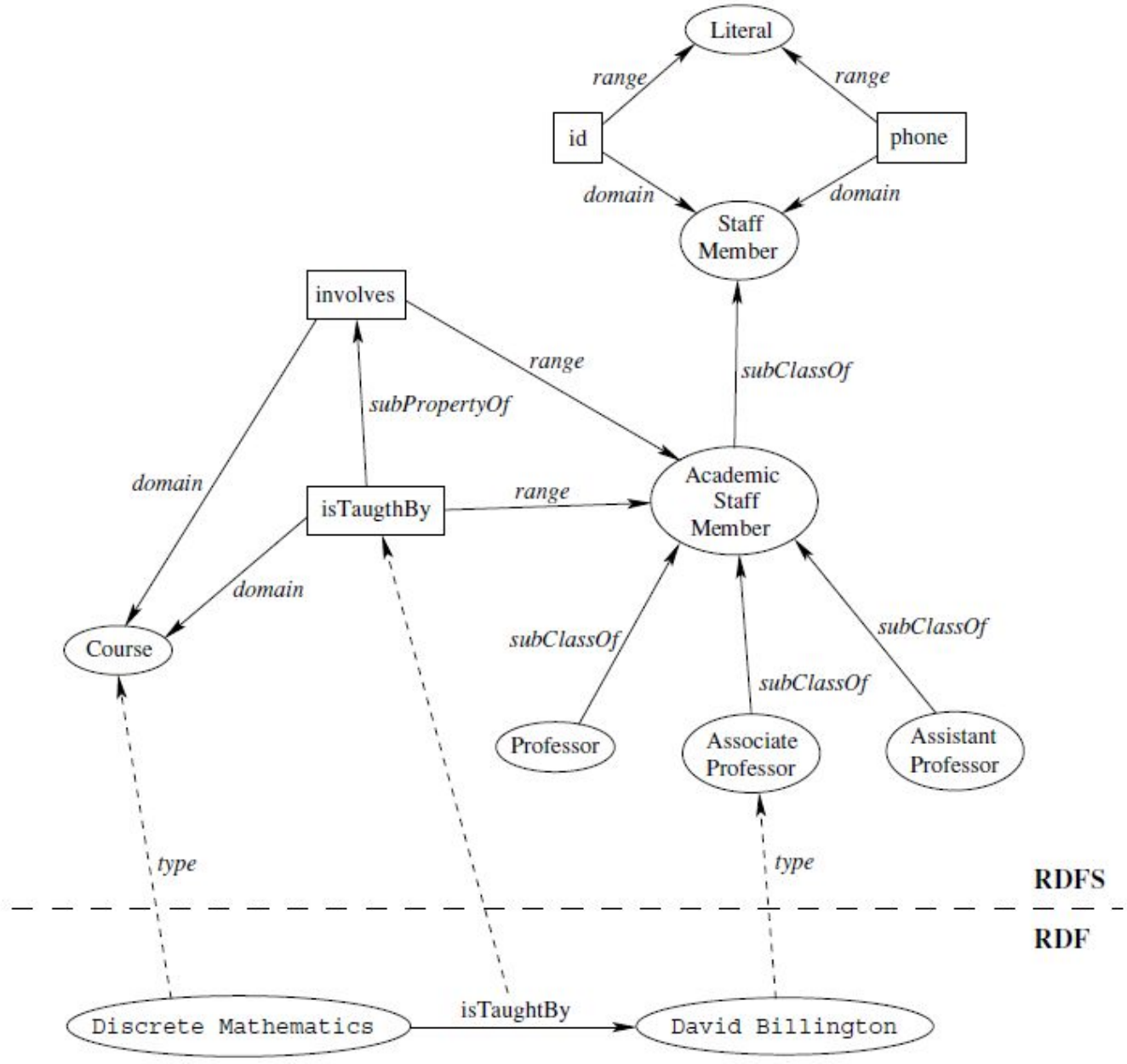
- Classes (*rdfs:Class*) and Properties (*rdfs:Property*)
- Class Hierarchies and Inheritance (*rdfs:subClassOf*)
- Property Hierarchies (*rdfs:subPropertyOf*)
- Domain (*rdfs:domain*) and range (*rdfs:range*) of properties

It is similar to the object-oriented programming (OOP) paradigm with the difference that in OOP the central notion is the class (and properties are defined for them), while in RDF the central notion is the property and classes are used to specify their domain/range.

Classes and instances

Individual objects that belong to a class are referred to as **instances** of that class (*rdf:type*).

Graphical example



RDF example

```
<rdfs:Class rdf:about="#lecturer">
        <rdfs:subClassOf rdf:resource="#staffMember"/>
</rdfs:Class>
<rdf:Property rdf:ID="phone">
        <rdfs:domain rdf:resource="#staffMember"/>
        <rdfs:range
rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

Utility properties

- **rdfs:seeAlso** relates a resource to another resource that explains it
- **rdfs:isDefinedBy** is a subproperty of rdfs:seeAlso and relates a resource to the place where its definition, typically an RDF schema, is found
- **rdfs:comment** support comments that can be associated with a resource
- **rdfs:label** is a human-friendly name associated with a resource

```
<rdfs:Class rdf:ID="course">
        <rdfs:comment>The class of courses</rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:ID="isTaughtBy">
        <rdfs:comment>
            Inherits its domain ("course") and range ("lecturer")
            from its superproperty "involves"
        </rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="#involves"/>
</rdf:Property>
```

Reasoning

Inference system

Sound and complete set of inference rules:

- The RDF inference system consists of **inference rules**
- **Sound:** inference rules prove only formulas that are valid with respect to its semantics
- **Complete:** every formula having a certain property can be derived) inference systems

Examples of rules:

(transitivity)
IF E contains the triples (?u,rdfs:subClassOf,?v) and (?v,rdfs:subclassOf,?w)
THEN E also contains the triple (?u,rdfs:subClassOf,?w)

(inheritance)
IF E contains the triples (?x,rdf:type,?u) and (?u,rdfs:subClassOf,?v)
THEN E also contains the triple (?x,rdf:type,?v)

RDF Inferencing by example

Type (rdf:type) propagation through rdfs:subClassOf

:Fausto Giunchiglia	rdf:type	:Professor
:Professor	rdfs:subClassOf	:Faculty
:Fausto Giunchiglia (inferred)	rdf:type	:Faculty

Relationship propagation through rdfs:subPropertOf

:professorshipAt	rdfs:subProperytOf	:affiliationWith
:Fausto Giunchiglia	:professorshipAt	:UniTN
:Fausto Giunchiglia	:affiliationWith	:UniTN (inferred)

Type identification through rdfs:domain

:professorshipAt	rdfs:domain	:Person
:Fausto Giunchiglia	:professrshipAt	:UniTn
:Fausto Giunchiglia (inferred)	rdf:type	:Person

RDF Inferencing by example

Type identification through rdfs:range

:professorshipAt	rdfs:range	:Educational_Institution
:Fausto_Giunchiglia	:professrshipAt	:UniTn
:UniTn	rdf:type	:Educational_Institution (inferred)

Inferencing through rdfs:domain and rdfs:subClassOf

:Researcher	rdfs:subClassOf	:Scientist
:hIndex	rdfs:domain	:Researcher
:Fausto Giunchiglia	:hIndex	44
:Fausto Giunchiglia	rdf:type	:Researcher (inferred)
:Fausto Giunchiglia	rdf:type	:Scientist (inferred)

Inferencing through rdfs:range and rdfs:subClassOf

:Educational_Institution	rdfs:subClassOf	:Organization
:professorshipAt	rdfs:range	: Educational Institution
:Fausto Giunchiglia	:professorshipAt	:UniTn
:UniTn	rdf:type	:Educational Institution (inferred)
:UniTn	rdf:type	:Organization (inferred)

Intersection and union in RDF

Set Intersection (if an entity e is in X, it is also in both Y and Z)

X	rdfs:subClassOf	Y
X	rdfs:subClassOf	Z
e	rdf:type	X
e	rdf:type	Y (inferred)
e	rdf:type	Z (inferred)

Set Union (any entity e that belongs either to Y or Z also belongs to X)

Y	rdfs:subClassOf	X	
Z	rdfs:subClassOf	X	
e	rdf:type	Y	or
e	rdf:type	Z	
e	rdf:type	X (inferred)	

Summary

Summary

- RDF provides a foundation for representing and processing metadata
- RDF has a graph-based data model
- RDF has a (XML-based) syntax and a semantics (via RDF Schema)
- RDF has a decentralized philosophy and allows incremental building of knowledge, and its sharing and reuse across the Web

- RDF is domain-independent
- RDF Schema provides a mechanism for describing specific domains
- RDF Schema is a primitive ontology language
- It offers certain modelling primitives with fixed meaning

Exercises

Exercise 1

Produce an RDF triple representation of the product, manufacturer and stock information provided in the following table.

ID	Model Number	Division	Product Line	Manufacturing Location	SKU	Available
1	RT-11	Safety	Safety valve	Trento	LM5647	70
2	RTX-56	Safety	Safety valve	Trento	DK3852	30
3	MBB-32	Accessories	Monitor	Hong Kong	CM7823	50
4	DR-43	Control Engineering	Sensor	Malaysia	SN2643	30

Table: Product

Solution

Subject	Predicate	Object
product:Product1	product:id	1
product:Product1	product:modelNumber	RT-11
product:Product1	product:division	Safety
product:Product1	product:productLine	Safety Valve
product:Product1	product:manufacturingLocation	Trento
product:Product1	product:sku	LM5647
product:Product1	product:available	70
product:Product2	product:id	2
product:Product2	product:modelNumber	RTX-56
…		

Exercise 2

Applications that use RDF data from multiple sources need to overcome the issue of managing terminology.

Suppose that one source uses the term *analyst* and another one uses the term *researcher*. How can you represent the fact that:

2.1) *researcher* is a special case of *analyst*?

2.2) *researcher* and *analyst* may overlap?

2.3) *researcher* and *analyst* are equivalent?

Solution

2.1) If a researcher is a special case of analyst, then all researchers are also analysts. This kind of "if/then" relationship can be represented with a single rdfs:subClassOf relation.

:Researcher rdfs:subClassOf :Analyst

2.2) In this case we can define a new class and express the fact that both classes specialize it (so they may overlap).

:Researcher rdfs:subClassOf :Investigator

:Analyst rdfs:subClassOf :Investigator

2.3) RDFS does not provide a primitive construct for expressing class equivalence. However, it can be represented using rdfs:subClassOf.

:Analyst rdfs:subClassOf

Exercise 3

Model the following problem in RDF:

"A military mission planner wants to determine off-limits areas, i.e. areas that cannot be targeted by weapons. There are two sources of information contributing to the decision. One source says that civilian facilities (e.g. churches, schools and hospitals) must never be targeted. Another source provides information about off-limits airspaces, called no-fly zones."

Solution

source1:CivilianFacility rdfs:subClassOf mmp:OffLimits

source2:NoFlyZone rdfs:subClassOf mmp:OffLimits

Exercise 4

Suppose an application imports RDF data from an excel file.

- There are two classes of entities, Person and Movie, defined by the import.
- For Person a property called *personName* is defined that gives the name by which that person is known.
- For Movie, the property called *movieTitle* gives the title under which the movie was released.

How to use the standard property *rdfs:label* to develop a generic display mechanism for showing both the names of the persons and titles of the movies?

Solution

We can define each of the properties as subproperty of rdfs:label

personName rdfs:subPropertyOf rdfs:label
movieTitle rdfs:subPropertyOf rdfs:label

Exercise 5a

Consider that a shipping company has a fleet of vessels including:

- new ones that are under construction
- old ones that are being repaired
- the ones that are currently in service
- the ones that have been retired from service

Represent information in the table in RDF

Name	Maiden Voyage	Next Departure	Decommission Date	Destruction Date
Titanic	April 10, 1912			April 14, 1912
MV 16	May 23, 2001	November 29, 2013		
MV 22	June 8, 1970		February 10, 1998	

Table: Ships

Solution

RDF statements to be produced include:

ship:Titanic ship:destructionDate "April 14, 1912"

ship:MV16 ship:nextDeparture "November 29, 2013"

ship:MV22 ship:maidenVoyage "June 8, 1970"

The following statements hold between classes:

ship:DeployedVessel rdfs:subClassOf ship:Vessel

ship:InServiceVessel rdfs:subClassOf ship:Vessel

ship:OutOfServiceVessel rdfs:subClassOf ship:Vessel

Exercise 5b

How can we represent the following inferences?:

- **if a vessel has a maiden voyage, then it is a Deployed Vessel**
- **if next departure date is set, then it is a In Service Vessel**
- **if it has decommission date or destruction date, then it is a Out Of Service Vessel**

Solution

ship:maidenVoyage ship:DeployedVessel	rdfs:domain
ship:nextDeparture ship:InServiceVessel	rdfs:domain
ship:decommisionDate ship:OutOfServiceVessel	rdfs:domain
ship:destructionDate ship:OutOfServiceVessel	rdfs:domain

Exercise 6

In the table below we can see that the ships have commanders. How can we assert that the commander of a ship is a captain? And that John and Alexander are therefore two captains?

Name	Maiden Voyage	Next Departure	Commander
MV 16	May 23, 2001	November 29, 2013	John
MV 22	June 8, 1970		Alexander

Table: Ships

Solution

ship:hasCommander	rdfs:range	ship:Captain
ship:John	rdf:type	ship:Captain
ship:Alexander	rdf:type	ship:Captain

Introduction to OWL

Requirements for Ontology Languages

Ontology languages allow users to write explicit, formal conceptualizations of domain models (i.e. formal ontologies)

The main requirements are:

- A well-defined formal syntax
- Sufficient expressive power
- Convenience of expression
- Formal semantics
- Support for efficient reasoning
- A good tread-off between expressivity and efficiency

OWL (Web Ontology Language) has been designed to meet these requirements for the specification of ontologies and to reason about them and their instances

Reasoning capabilities required

Class membership

If x is an instance of a class C, and C is a subclass of D, then we can infer that x is an instance of D

Equivalence of classes

If class A is equivalent to class B, and class B is equivalent to class C, then A is equivalent to C

Disjointness and Consistency

Determine that if the classes A and B are disjoint there cannot be individuals x which are instances of both A and B. This is an indication of an error in the ontology.

Classification

Certain property-value pairs are a sufficient conditions for membership in a class A; if an individual x satisfies such conditions, we can conclude that x must be an instance of A.

Limitations in the expressive power of RDF schema

Range restrictions

We cannot declare range restrictions that apply to some classes only (e.g. cows eat only plants, while other animals may eat meat too).

Disjointness of classes

We cannot declare that two classes are disjoint (e.g. male and female).

Combinations of classes

We cannot define new classes as union, intersection, and complement of other classes (e.g. person is the disjoint union of the classes male and female).

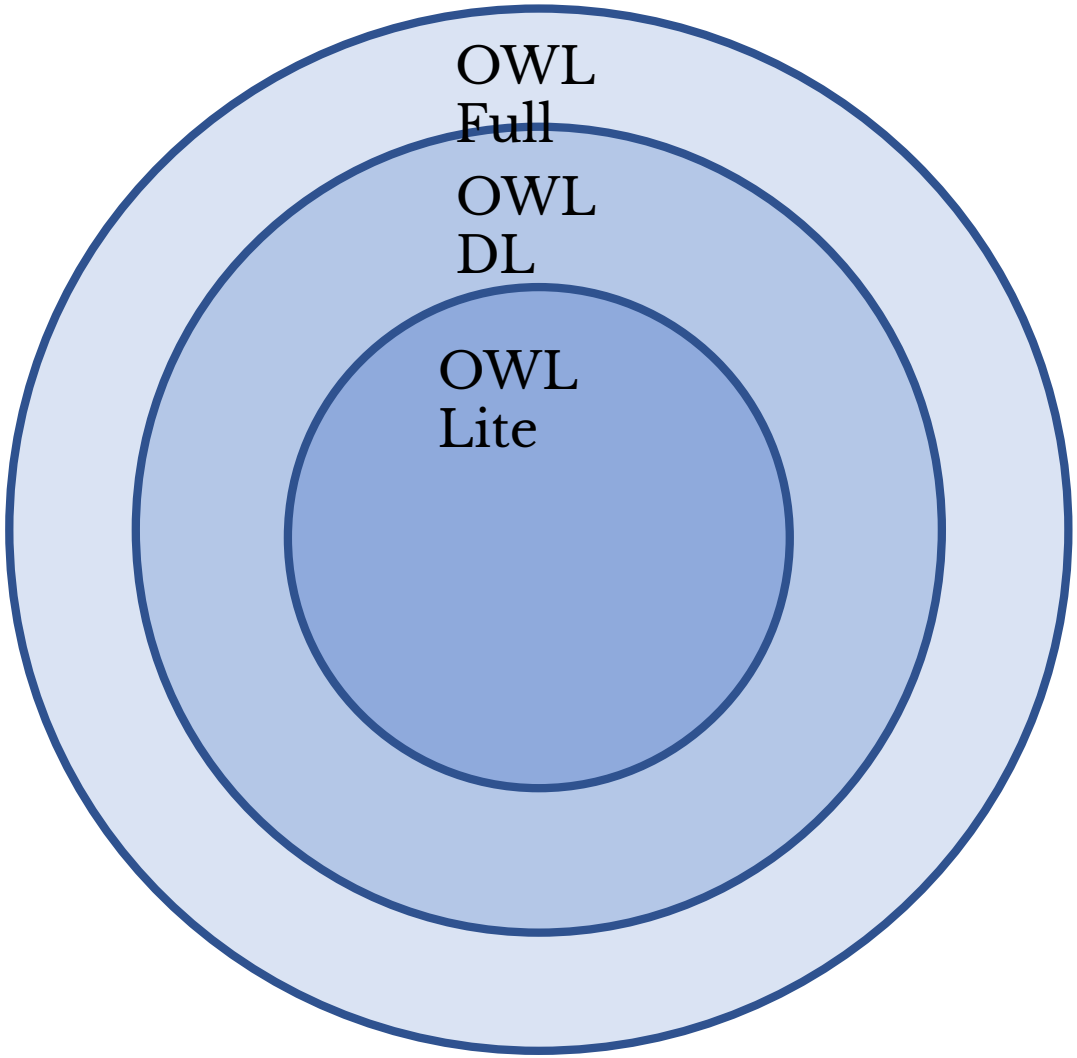
Cardinality restrictions

We cannot express restrictions in the number of relations (e.g. a person has exactly two parents, a course is taught by at least one lecturer)

Meta-properties

Transitive property (e.g. “greater than”)
Unique property (e.g. “is mother of”)
Inverse property (e.g. “eats” and “is eaten by”)

OWL sub-languages



- Each OWL Lite representation belongs to OWL DL
- Each OWL DL representation belongs to OWL Full
- Each valid OWL Lite conclusion is also valid in OWL DL
- Each valid OWL DL conclusion is also valid in OWL Full

OWL sub-languages

OWL Lite trades expressivity for efficiency

- The less expressive of all languages (it cannot be used to express enumerated classes, disjointness, and arbitrary cardinality restrictions)
- It allows assigning simple cardinality constraints of kind 0 or 1
- It allows encoding simple classification hierarchies (e.g., taxonomies and thesauri)
- Partially compatible with RDF

OWL DL is a balance between expressivity and computational completeness

- More expressive than OWL Lite while guarantees decidability
- It allows expressing all DL constructs, some of them with certain restrictions (e.g. the restriction of not making a class an instance of another class)
- Partially compatible with RDF

OWL Full trades computational completeness for expressivity

- More expressive than OWL DL, maximum expressiveness (e.g., a class can be represented also as an individual)
- It is computationally very expensive and does not guarantee decidability
- Fully upward-compatible with RDF, both syntactically and

The OWL Full language

OWL XML/RDF syntax

HEADER

```
<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/XLMSchema#">
```

ONTOLOGY

```
<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>

</rdf:RDF>
```

Classes

Defined using owl:Class that is a subclass of rdfs:Class
owl:Thing is the most general class, which contains everything
owl:Nothing is the empty class

DISJOINT CLASSES *owl:disjointWith*

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

EQUIVALENT CLASSES *equivalentClass*

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

Properties

Data type properties relate objects to datatype values (ATTRIBUTES)

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

Object properties relate objects to other objects (RELATIONS)

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

Property restrictions: a kind of class description (I)

VALUE CONSTRAINT *owl:allValuesFrom*

A value constraint puts constraints on the range of the property. It corresponds to universal quantification.

```
<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>
```

Property restrictions: a kind of class description (I)

CARDINALITY CONSTRAINT *someValuesFrom / owl:hasValue*

A cardinality constraint puts constraints on the number of values. It corresponds to the existential quantification or can indicate a specific value.

```
<owl:Class rdf:about="#firstYearCourse">
        <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#teaches"/>
                <owl:someValuesFrom
rdf:resource="#undergraduateCourse"/>
                (or)
                <owl:onProperty rdf:resource= "#isTaughtBy"/>
                <owl:hasValue rdf:resource= "#949352"/>
    </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Cardinality restrictions (I)

owl:maxCardinality

It describes a class of all individuals that have at most N semantically distinct values (individuals or data values) for the property.

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:maxCardinality
rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxCardinality>
</owl:Restriction>
```

owl:minCardinality

It describes a class of all individuals that have at least N semantically distinct values (individuals or data values) for the property.

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">2</owl:minCardinality>
```

Cardinality restrictions (I)

owl:cardinality

It describes a class of all individuals that have exactly N semantically distinct values (individuals or data values) for the property concerned, where N is the value of the cardinality constraint.

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:cardinality
rdf:datatype="&xsd;nonNegativeInteger">2</owl:cardinality>
</owl:Restriction>
```

This construct is redundant in that it can be replaced by a pair of matching owl:minCardinality and owl:maxCardinality constraints with the same value.

Meta-properties (I)

EQUIVALENCE *owl:equivalentProperty*

x P y implies x Q y

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

NOTE: in RDF we need P rdfs:subPropertyOf Q and Q rdfs:subPropertyOf P

INVERSE *owl:inverseOf*

x P y implies y Q x

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

Meta-properties (II)

SYMMETRIC *owl:SymmetricProperty*

x P y implies y P x

```
<owl:ObjectProperty rdf:ID="married">
        <rdf:type rdf:resource="&owl;SymmetricProperty"/>
        <rdfs:range rdf:resource="#person"/>
        <rdfs:domain rdf:resource= "#person"/>
</owl:ObjectProperty>
```

TRANSITIVE *owl:TransitiveProperty*

x P y and y P z implies x P z

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

Functional and inverse functional properties

FUNCTIONAL PROPERTY *owl:FunctionalProperty*

A functional property is a property that can have only one value as range for any given individual (e.g., hasMother , hasPresident).

INVERSE FUNCTIONAL PROPERTY *owl:InverseFunctionalProperty*

It defines a property that cannot have the same value as range for any given individual (e.g., MotherOf , PresidentOf).

Enumerations

It allows a class to be defined by exhaustively enumerating its instances. The class extension of a class described with *owl:oneOf* contains exactly the enumerated individuals, no more, no less.

```
<owl:Class rdf:ID="weekdays">
        <owl:oneOf rdf:parseType="Collection">
                <owl:Thing rdf:about="#Monday"/>
                <owl:Thing rdf:about="#Tuesday"/>
                <owl:Thing rdf:about="#Wednesday"/>
                <owl:Thing rdf:about="#Thursday"/>
                <owl:Thing rdf:about="#Friday"/>
                <owl:Thing rdf:about="#Saturday"/>
                <owl:Thing rdf:about="#Sunday"/>
        </owl:oneOf>
</owl:Class>
```

Intersection

```
<owl:Class>
 <owl:intersectionOf rdf:parseType="Collection">
  <owl:Class>
   <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Tosca" />
    <owl:Thing rdf:about="#Salome" />
   </owl:oneOf>
  </owl:Class>
  <owl:Class>
   <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Turandot" />
    <owl:Thing rdf:about="#Tosca" />
   </owl:oneOf>
  </owl:Class>
 </owl:intersectionOf>
</owl:Class>
```

Union

```
<owl:Class>
 <owl:unionOf rdf:parseType="Collection">
  <owl:Class>
   <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Tosca" />
    <owl:Thing rdf:about="#Salome" />
   </owl:oneOf>
  </owl:Class>
  <owl:Class>
   <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Turandot" />
    <owl:Thing rdf:about="#Tosca" />
   </owl:oneOf>
  </owl:Class>
 </owl:unionOf>
</owl:Class>
```

Complement

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

Instances

Instances of classes are declared as in RDF:

```
<rdf:Description rdf:ID="949352">
        <rdf:type rdf:resource= "#academicStaffMember"/>
</rdf:Description>
<academicStaffMember rdf:ID="949352">
        <uni:age rdf:datatype="&xsd;integer">39<uni:age>
</academicStaffMember>
```

Same instances:

```
<rdf:Description rdf:about="#William_Jefferson_Clinton">
  <owl:sameAs rdf:resource="#BillClinton"/>
</rdf:Description>
```

Different instances:

```
<Opera rdf:ID="Nozze_di_Figaro">
  <owl:differentFrom rdf:resource="#Don_Giovanni"/>
</Opera>
```

OWL DL and OWL lite

OWL DL

OWL DL is a sublanguage of OWL which places a number of constraints on the use of the OWL language constructs which ensure that computational complexity is the same as corresponding Description Logic.

- Each individual must be an instance of a class, and in particular of owl:Thing
- Pairwise separation between classes, datatypes, datatype properties, object properties, annotation properties, ontology properties (i.e., the import and versioning stuff), individuals, data values and the built-in vocabulary. This means that, for example, a class cannot be at the same time an individual.
- No cardinality constraints can be placed on transitive properties or their inverses or any of their super-properties.
- It is allowed to use the intersectionOf construct with any number of classes and of any non negative integer in the cardinality restrictions value fields

OWL lite

OWL lite is a sublanguage of OWL DL which places further constraints on the use of the OWL language constructs which ensure a lower computational complexity

- Users are allowed to use a subset of the OWL, RDF and RDFS vocabulary
- To define a class, one must use the OWL construct owl:Class
- OWL constructs complementOf, disjointWith, hasValue, oneOf and unionOf are not allowed
- All three cardinality constructs – cardinality, maxCardinality and minCardinality, can only have 0 or 1 in their value fields
- equivalentClass and intersectionOf cannot be used in a triple if the subject or object represents an anonymous class

Exercises

Exercise 1

Suppose that a family consists of a father (John), a mother (Maria), two sisters (Sara and Jenifer) and two brothers (David and Robert). In an OWL representation the two brothers and the two sisters are codified as follows:

```
:David  :hasFather   :John
:Sara   :hasFather   :John
:John   :spouseOf    :Maria
```

Later on another property :hasChild is codified.

(i) What will be the output of the following SPARQL Query when a reasoner is activated?

```
:John          :hasChild          ?y
```

Exercise 1 (cont)

(ii) Expand the OWL representation in a way that supports returning non-empty result of the following query and this expansion is independent of the entity-entity triples.

:John :hasChild ?y

(iii) Add also the following axioms to the dataset.

:Jenifer :hasFather :John

:Robert :hasFather :John

What results the following query will return?

:John :hasChild ?y

(iv) How can we infer the spouse relation in the reverse direction?

Solution I

(i) The result of the query is empty.

(ii) We can make the property :hasFather as an inverse property of :hasChild

:hasFather owl:inverseOf :hasChild

Query Result:

:David

:Sara

(iii)

:David

:Sara

:Jenifer

:Robert

(iv) We can make the relation :spouseOf its own inverse as follows:

:spouseOf owl:inverseOf :spouseOf

Exercise 2

Within a family, the following relations are applicable in both directions (from subject to object, and vice versa):

:spouseOf

:marriedTo

:siblingOf

whereas the same those not always apply to the following:

:brotherOf

:sisterOf

(i) Which property holds in the relations that are applicable in both directions?

(ii) How can we represent these relations in OWL?

(iii) In which basic category this property belongs?

Solution 2

(i) Symmetric property

(ii)
```
:spouseOf          rdf:type
   owl:SymmetricProperty
:marriedTo         rdf:type
   owl:SymmetricProperty
:siblingOf         rdf:type
   owl:SymmetricProperty
```

(iii) The symmetric property is an object property. Moreover, the domain and range of the symmetric property are the same (owl:Class)

Exercise 3

Consider that in the family of John and Maria, also John's father (James) and mother (Jerry) live. Relations such as :hasAncestor and :hasDescendent can be applied between different levels. For example:

:John :hasAncestor :James
:Sara :hasAncestor :John
:James :hasDescendent :John
:John :hasDescendent :Sara

(i) Which property holds in the relations that are applicable in different levels of the hierarchy?

(ii) How can we represent these relations in OWL?

(iii) In which basic category this property belongs?

(iv) Show the results of the following queries:

a) :James :hasDescendent ?y

b) :John :hasAncestor ?y

Solution 3

(i) Transitive property

(ii) :hasAncestor rdf:type owl:TransitiveProperty
:hasDescendent rdf:type owl:TransitiveProperty

(iii) The transitive property is an object property.

(iv) a) :John
:Sara

b) :James

Exercise 4

(i) In RDFS we can represent that two classes :Test and :Experiment are equivalent.

:Test rdfs:subClassOf :Experiment

:Experiment rdfs:subClassOf :Test

Convert this representation in OWL.

(ii) In RDFS we can represent that two properties :hasChild and :hasKid are equivalent.

:hasChild rdfs:subPropertyOf :hasKid

:hasKid rdfs:subPropertyOf :hasChild

Convert this representation in OWL.

(iii) Is there any way to represent the fact that two entities (or individuals) :Italia and :Il_Bel_Paese are the same?

Solution 4

(i) :Test owl:equivalentClass :Experiment

(ii) :hasChild owl:equivalentProperty :hasKid

(iii) :Italia owl:sameAs :Il_Bel_Paese

Exercise 5

(i) Which OWL property allows to have exactly one value for a particular individual?

(ii) The following relations can be defined using the OWL property above.

:hasFather

:hasMother

Represent them in OWL and demonstrate their use with necessary entity-entity axioms.

Solution 5

(i) OWL Functional property

(ii)

:hasFather rdf:type owl:FunctionalProperty

:hasMother rdf:type owl:FunctionalProperty

Two entity-entity axioms are provided below:

:John :hasFather :James

:John :hasFather :Handler

The objects :James and :Handler are the values of the same subject and property. We already have defined that :hasFather property is functional. Therefore, it can be concluded that :James and :Handler refer to the same person.

Exercise 6

(i) Which OWL property allows to have exactly one value for a particular object?

(ii) Demonstrate the use of such a property in developing applications such as the detection of possible duplicates.

Solution 6

(i) OWL Inverse Functional property

(ii) We can encode the property :SSN (social security number) as follows:

:SSN rdf:type owl:InverseFunctionalProperty

Two entity-entity axioms are provided below:

mo:James :SSN N123812834
ps:Handler :SSN N123812834

The subjects :James and :Handler are attached to the same social security number, which cannot be shared by two different persons. Therefore, we can conclude that mo:James and ps:Handler are the same entity.

References

- RDF Primer (W3C): http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/
- Resource Description Framework (RDF): Concepts and Abstract Syntax (W3C): http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
- RDF Schema (W3C): http://www.w3.org/TR/rdf-schema/
- G. Antoniou & F. van Harmelen (2004). A Semantic Web Primer (Cooperative Information Systems). MIT Press, Cambridge MA, USA.
- F. Giunchiglia, F. Farazi, L. Tanca, and R. D. Virgilio. The semantic web languages. In Semantic Web Information management, a model based perspective. Roberto de Virgilio, Fausto Giunchiglia, Letizia Tanca (Eds.), Springer, 2009.
- D. Allemang and J. Hendler. Semantic web for the working ontologist: modeling in RDF, RDFS and OWL. Morgan Kaufmann Elsevier, Amsterdam, NL, 2008.
- OWL Web Ontology Language(W3C): http://www.w3.org/TR/2004/REC-owl-ref-20040210/

- G. Antoniou & F. van Harmelen (2004). A Semantic Web Primer (Cooperative Information Systems). MIT Press, Cambridge MA, USA.
- D. Allemang and J. Hendler. Semantic web for the working ontologist: modeling in RDF, RDFS and OWL. Morgan Kaufmann Elsevier,