Semantic Data

Chapter 3 : The semantic web resource description framework

Jean-Louis Binot

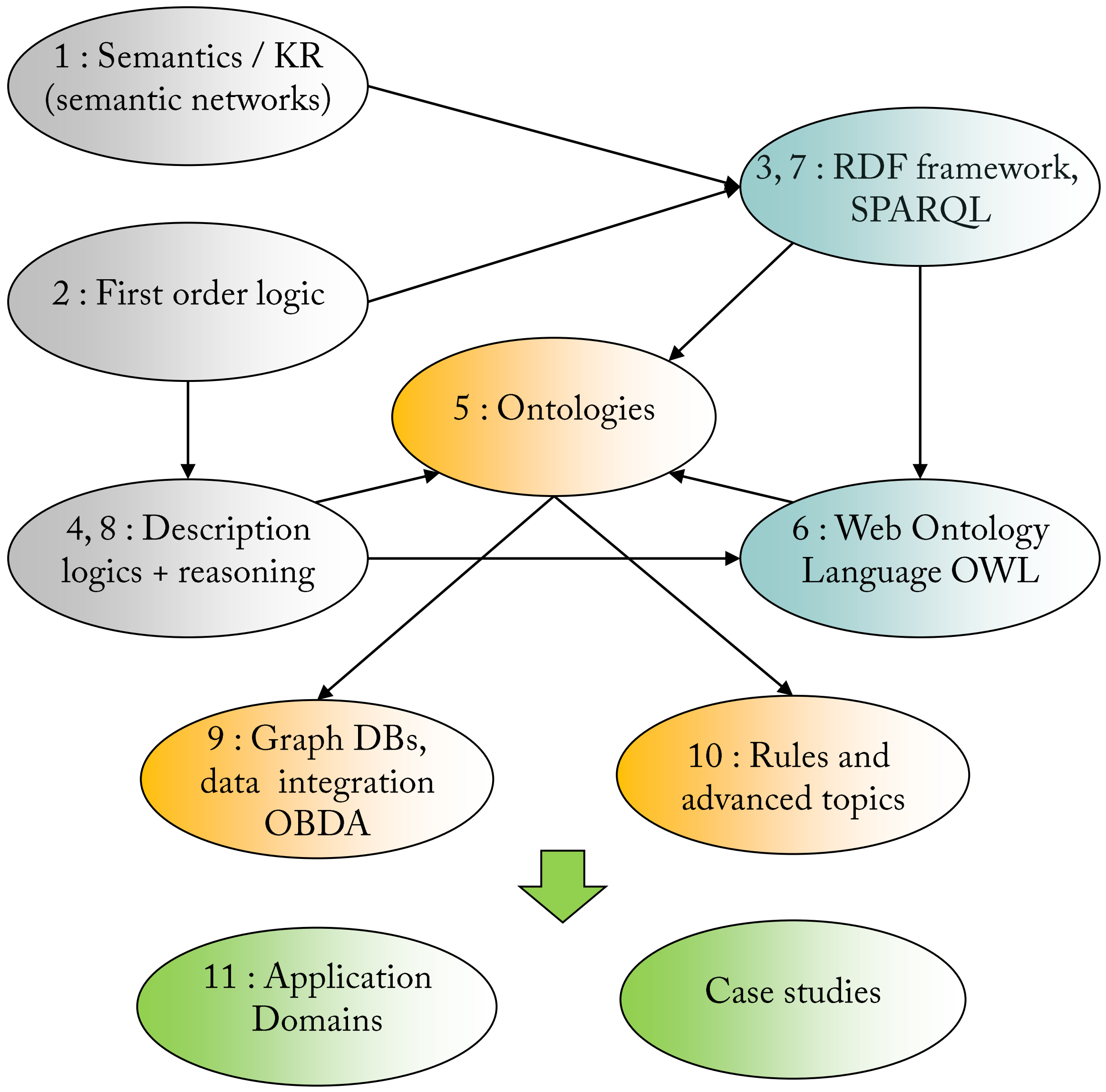
Course content outline

Credits : 5 (theory 25 h, practice 10 h, project 45 h)

Theory (25 h):

1. Semantics and knowledge representation.
2. Introduction to first order logic.
3. The semantic web resource description framework.
4. Description logics.
5. Ontologies and ontology engineering.
6. The Web Ontology Language : OWL.
7. Querying the semantic web : SPARQL.
8. Reasoning with description logics.
9. Data integration and ontology-based data access.
10. Rules and advanced topics.
11. Application domains for semantic data.

Case studies : real cases for genuine business customers; integrated in the relevant theory sessions.



Sources and recommended readings

- Theory : there are no additional references required for this chapter.
  - The students are expected to consult the W3C sources as needed for practice, and to understand XML as a basis for RDF; XML itself is not part of the required material.

- Sources and useful additional readings :
  - The RDF section is inspired from the W3C RDF Primer 1.1 (2014) and the RDF Primer (2004).
  - The RDFS section is inspired from the same sources and RDF Semantics, section 7.
  - Books : *Semantic Web Primer (Antoniou and van Harmelen 2004); Web Data Management* (*Abiteboul et al. 2011).*

- University courses having partially inspired ideas and examples for this chapter :
  - *Web sémantique, O. Corby, Inria.*
  - *Apprentissage symbolique et web sémantique, B. Amman, UPMC.*
  - *Semantic Web Technologies, H. Paulheim, Universität Manheim.*

Agenda

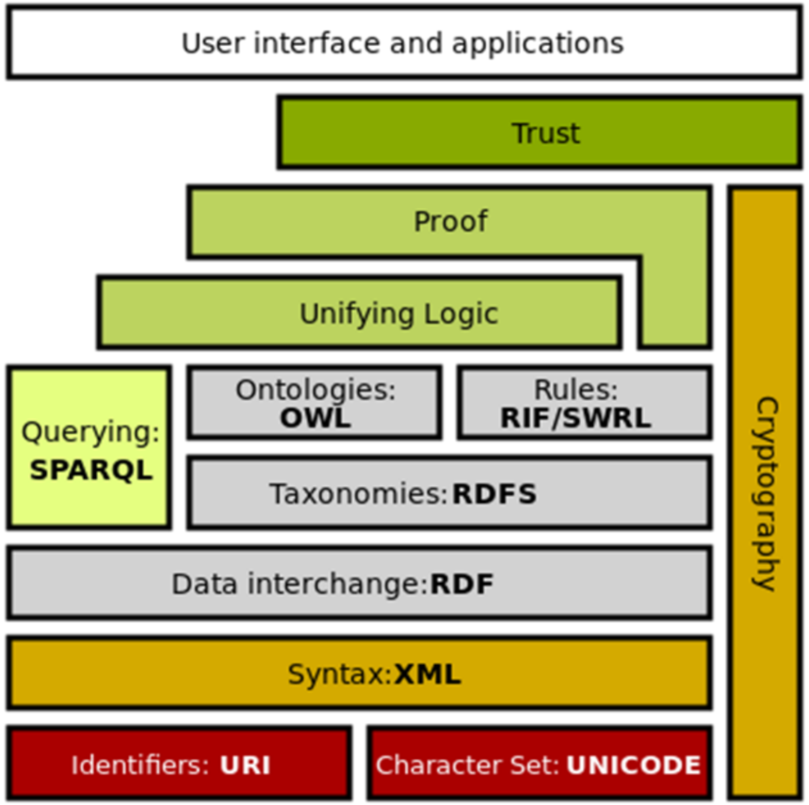
1	The semantic web stack
2	Why not HTML or XML ?
3	Resource description framework
4	RDF Schema

The semantic web standards (W3C consortium)

Multiple layers building on each other :

- RDF (Resource Description Framework) :
  - Semantic information expressed as triples.
- RDFS (Resource Description Framework Schema) :
  - Extension of RDF to support simple ontologies.
- OWL (Web Ontology Language) :
  - Representation of ontologies mapped on description logics.
- SPARQL (RDF Query Language).
- RIF (Rule Interchange Format) and SWRL (Semantic Web Rule Language).

The initial syntax is based on XML (other syntaxes have been added).



The semantic web stack of standards

Industrial relevance

- The dream of the semantic web was to have *many software agents that collect Web content from diverse sources, process the information and exchange the results with other programs … automated services (Berners-Lee et al., 2001).*

  That dream has not happened yet

- However, the underlying technologies are widely used :
  - Ontologies (large medical ontologies, others) use semantic web standards (OWL).
  - Knowledge graphs are typically based on RDF triples.
  - Rich content for search engines and portals, recommendation engines, personal assistants …
  - NoSQL graph databases : RDF stores.
  - RDF-based data integration.
  - …

Agenda

1	The semantic web stack
2	Why not HTML or XML ?
3	Resource description framework
4	RDF Schema

Why not just HTML ?

<h2>Nonmonotonic Reasoning: Context-
Dependent Reasoning</h2>
<i>by V. Marek and
M. Truszczynski</i>
Springer 1993
ISBN 0387976892

Nonmonotonic Reasoning: Context-Dependent Reasoning

by ***V. Marek*** *and* ***M. Truszczynski***
Springer 1993
ISBN 0387976892

(example from Antoniou and van Harmelen, 2004)

- HTML is oriented toward visual **presentation and** keyword **search**, targeting human users.
- However the semantic web is intended to be a web of linked data accessible by machines.
- We need a way to describe data content in a wide variety of fields from e-commerce to bioinformatics, and many others.
- HTML is not suited for that task.

Might XML be a better solution ?

Reminder :

- XML (eXtensible Markup Language) is a standard for data exchange on the Web :
  - Flexible, generic, platform independent;
  - Coming with a wide variety of tools (parsers, programming interfaces, manipulation languages);
  - Now an industry standard for data integration.
- An XML element consists of an opening tag, its content, and a closing tag:

  <lecturer>Jean-Louis Binot</lecturer>

  - Tag names can be chosen almost freely.
- Typing an XML document can be done with a Document Type Definition, or an XML Schema, richer and newer.

Could we use XML to link data and capture their semantics ?

Why not just XML : an example

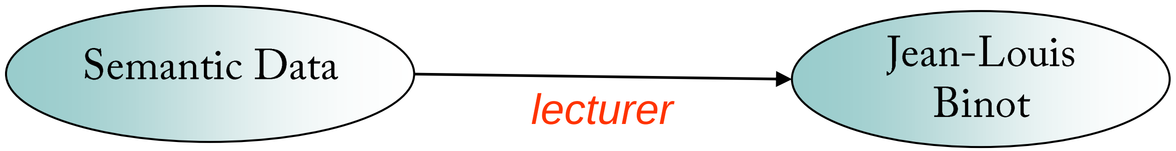
```
<course name= "Semantic Data">
    <lecturer>Jean-Louis Binot</lecturer>
</course>

<lecturer name= "Jean-Louis Binot">
    <teaches>Semantic Data</teaches>
</lecturer>

<teachingOffering>
    <lecturer> Jean-Louis Binot </lecturer>
    <course>Semantic Data</course>
</teachingOffering>
```

(adapted from Antoniou and van Harmelen 2004)

- The examples on the left correspond to the same relation (below) but show different ways to represent it in XML.
- If we want to query who is the lecturer of the course, it is impossible without knowing the specific tag structure used.
- An XML tag structure does not have any standard meaning :
  - The meaning is only assigned by the application using it.
  - XML has a syntax but no way to describe the semantics of the data !



Agenda

1. The semantic web stack
2. Why not HTML or XML ?
3. Resource description framework
4. RDF Schema

Basic ideas of the Resource Description Framework (RDF)

RDF aims at describing web resources. It relies on 3 fundamental concepts :

- Resources.
- Properties.
- Statements.

Resources

- Resources can represent anything we want to talk about or to refer to :
  - Web resources (documents, images, services, groups of resources…);
  - Individual entities or objects of the real world (humans, corporations, books…);
  - Abstract concepts (classes, relations, properties…).

- Each resource is identified by a Universal Resource Identifier or URI (RFC 3986) :
  - URLs (uniform resource locators) are a type of URI, providing explicit location of the resource :

    URI = scheme:[//authority]path[?query][#fragment]

    http://dbpedia.org/resource/Leonardo_da_Vinci

    http://www.example.org/bob#me

  - An URI can also be a universal resource name (URN) for a resource without location information.

    urn:isbn:0-486-27557-4 (a book ISBN – International Standard Book Number).

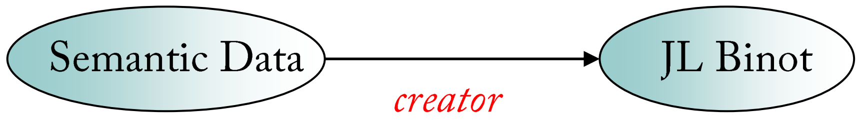
Properties

- Properties : special kind of resources, which describe relations between resources.
  - Examples : written by, age, title…
- Properties are also identified by URIs.
- Using URIs both for resources and properties has important advantages :
  - URIs are global identifiers : they provide a worldwide unique naming scheme.
  - Hence, they reduce the homonym problems arising with distributed data representation.

Statements

- Statements about resources take the form of triples <Subject Predicate Object>(*).
  - A statement can be interpreted as a logical (FOL) formula : *Predicate(Subject, Object)*.
- Triples can also express properties of resources : <Resource Property Value>.
  - This can be interpreted in FOL as *Property(Resource, Value)*.
- Abstract example: *the course Semantic Data has a creator who is JL Binot.*

Triple: *<Semantic Data> <creator> <JL Binot>*(*) .



* : *We use here <> to denote informally triple elements.*

The RDF data model is based on directed labelled graphs

- Abstract example :

<Bob> <is a> <person> .

<Bob> <is a friend of> <Alice> .

<Bob> <is born on> <14 July 1990> .

<Bob> <is interested in> <The Mona Lisa> .

<the Mona Lisa > <was created by> <Leonardo da Vinci> .

<the video "la Joconde in Washington"> <is about> < The Mona Lisa > .

- These triples form a labelled directed graph.

- It is a semantic network.

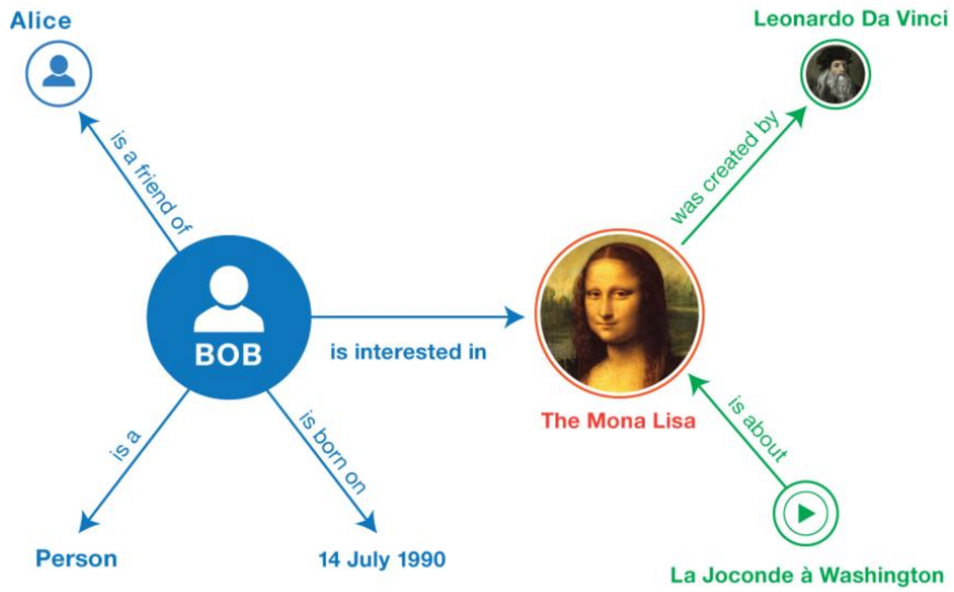


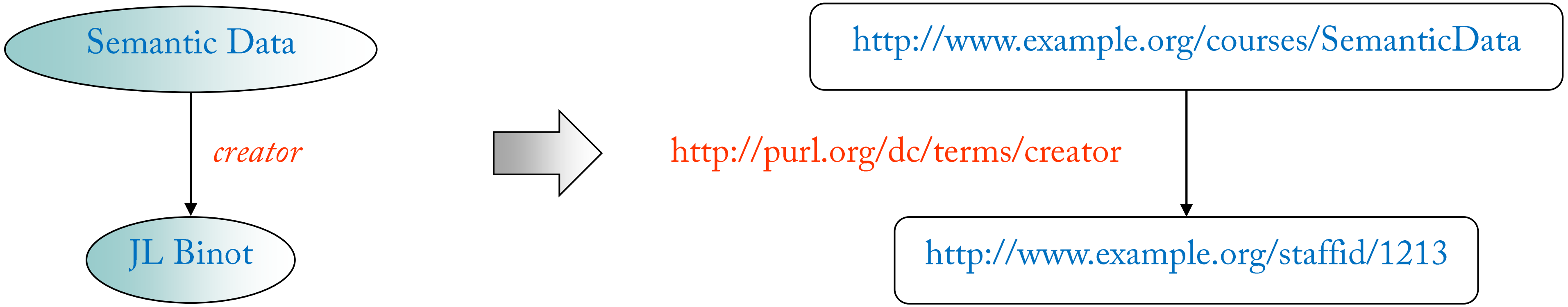
Fig. 1 Informal graph of the sample triples

(Source: RDF primer 1.1)

Triples with URIs

- Let us assume the following sources :
  - http://www.example.org describes an organisation teaching courses;
  - http://www.example.org/courses contains the resource descriptions of the courses;
  - http://www.example.org/staffid contains the resource descriptions of people;
  - http://purl.org/dc/terms/ the Dublin Core standard vocabulary for documents.

- Using explicit URIs, our abstract example would become :



RDF serialization - Turtle

- An RDF graph must be serialized to be stored into files.
  - Several syntaxes exist (cf. later in this chapter).

When not said otherwise, we will use the Turtle syntax (Terse RDF Triple language).

- Syntax of a basic RDF statement in Turtle :

  Subject Predicate Object . (do not forget the dot !)

  - In Turtle absolute URIs are enclosed between <>.

- Example from previous slide in Turtle :

  <http://www.example.org/courses/SemanticData> <http://purl.org/dc/terms/creator> <http://www.example.org/staffid/1213> .

  Still not really terse ! There are ways to make the syntax much more readable.

Namespaces and reference vocabularies

- RDF is agnostic about what URIs represent. URIs are given meaning by vocabularies.
- Vocabularies are organized by using namespaces :
  - A namespace is a set of unique symbols used to organize the names of objects.
  - The structure and meaning of the names can be fixed within the namespace.
  - Namespaces can be used for disambiguation : within a namespace, a name is always unique.
  - The namespace used is indicated by a prefix abbreviation, which shortens the URI notation.
- Ontologies defining vocabularies for specific domains can be published as standard namespaces.

  Example using the Dublin Core Metadata, a reference ontology for digital documents.

<http://purl.org/dc/terms/creator> → *PREFIX dc: <http://purl.org/dc/terms/>*
…
dc:creator

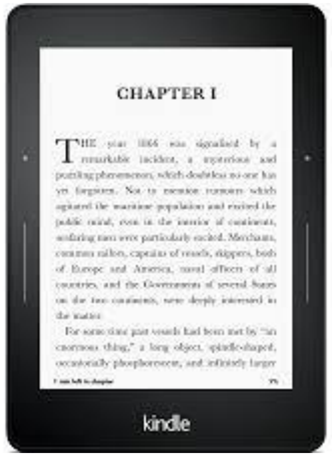
Dublin Core Metadata Initiative

- ISO Standard 15836; 15 vocabulary terms; a small but worldwide used taxonomy :

contributor	format	rights
coverage	identifier	source
creator	language	subject
date	publisher	title
description	relation	type

- Can be embedded into HTML; provides (among others) metadata for all ebooks.

```
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/" >
  <dc:identifier id="BookId" opf:scheme="ISBN">2-290-33529-0</dc:identifier>
  <dc:title>Hamlet</dc:title>
  <dc:language>fr</dc:language>
  <dc:creator>William Shakespeare</dc:creator>
  <dc:date>2004</dc:date>
</metadata>
```



Some usual vocabularies

- Some well-know vocabularies used in ontology contexts :

PREFIX dcterms:	<http://purl.org/dc/terms/>	Dublin Core Metadata : digital documents ontology.
PREFIX foaf:	<http://xmlns.com/foaf/0.1/>	Friend Of A Friend : ontology describing people and their relations.
PREFIX dbpedia:	<http://dbpedia.org/resource/>	DBPedia : ontology of structured data extracted from Wikipedia.
PREFIX skos:	<http://www.w3.org/2004/02/skos/core#>	Simple Knowledge Organisation System : a simplified ontology.

- Abbreviations for W3C standards:

PREFIX rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#>	Namespace for RDF.
PREFIX rdfs:	<http://www.w3.org/2000/01/rdf-schema#>	Namespace for RDFS.
PREFIX owl:	<http://www.w3.org/2002/07/owl>	Namespace for OWL.

Namespaces ./.

- http://www.example.org is a “second-level” domain name reserved for documentation purposes. It can be used to make examples in textbooks.

- We will use ex: as the corresponding namespace abbreviation :

  PREFIX ex: <http://www.example.org>

- Other variations on the “example” prefix can also be used, for instance :

  PREFIX exterms: <http://www.example.org/terms/> (for terms used in an example organisation)

  PREFIX exstaff: <http://www.example.org/staffid/> (for staff within the organisation)

- A default name space can be specified (for any namespace) :

  PREFIX : <http://example.org/>

Types of triple elements

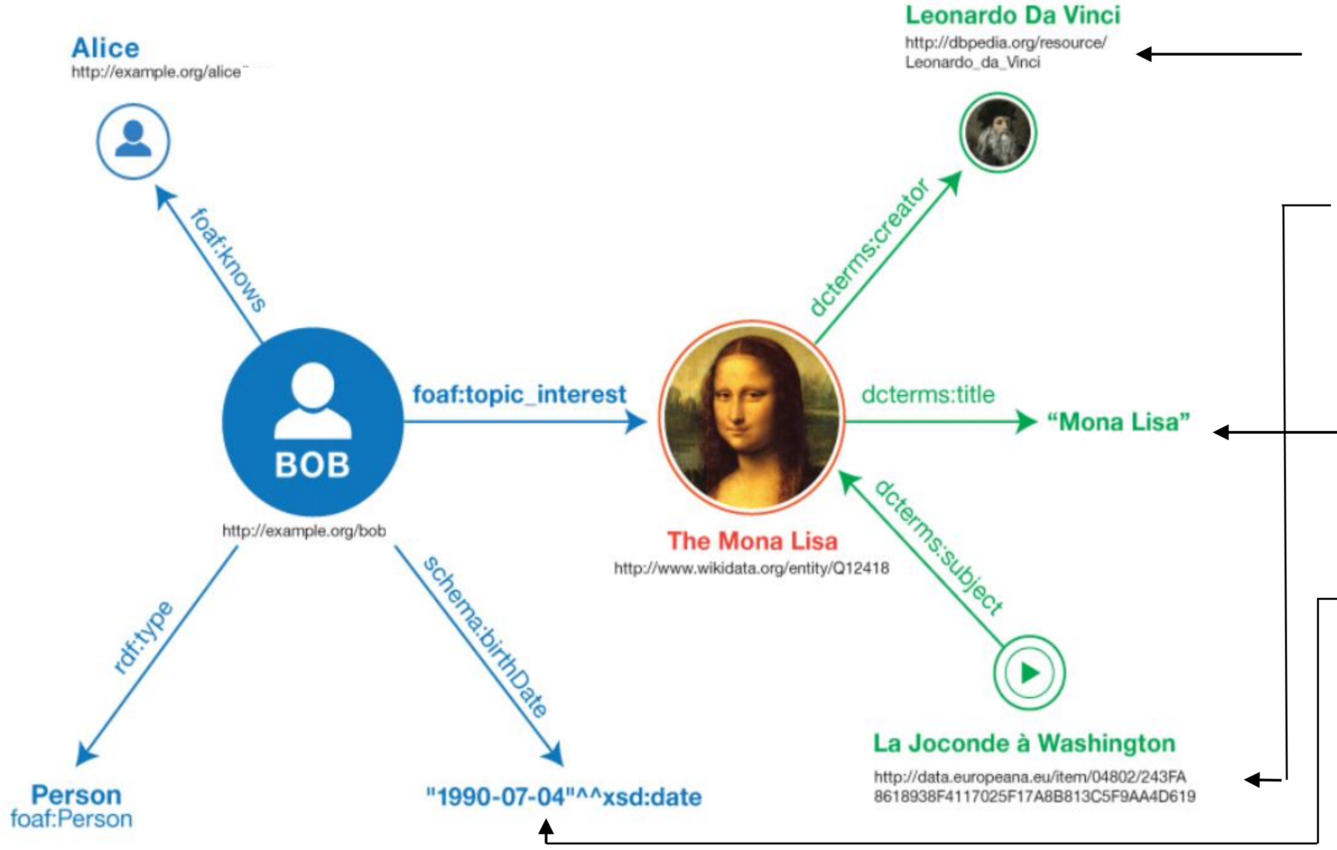
- Three types of elements can be used in a triple :
  - URIs or Universal Resource Identifiers;
  - Literals : basic values which are not URIs;
  - Blank nodes : can stand for an object without identifying it (like a variable).

- Their use is subject to the following limitations :
  - URIs can be used in any position of a triple;
  - Literals can only be used in object position;
  - Blank nodes only in subject or object position.

  Why not blank nodes in predicate position ?

	Subject	Predicate	Object
URI	✓	✓	✓
Literal	✗	✗	✓
Blank Node	✓	✗	✓

Example with URIs and literals



1. The URI for Leonardo da Vinci in Dbpedia.
2. The URI for a video about the Mona Lisa in Europeana.
3. "Mona Lisa" is a literal of type string (in Turtle, specifying the type string is not mandatory).
4. *"1990-07-04"^^xsd:date* is a literal of type date.

The notation *^^xsd:* introduces a type following the conventions of XML Schema datatypes.

Example 1 (source: RDF primer 1.1)

Literals and datatypes

- Literals are used for values such as strings, numbers, and dates.

- A literal in an RDF graph consists of two or three elements :
  - A lexical form, being a Unicode string;
  - A datatype URI, which determines how the lexical form maps to a literal value, and
  - Iff the datatype is langString (a language string), a non-empty language tag introduced by @.

  Concrete syntaxes may omit the string datatype for literals of type string.

  Most concrete syntaxes also represent language-tagged strings without the langstring datatype.

- Examples :
  - *"1990-07-04"^^xsd:date*
  - *"That Seventies Show"@en*

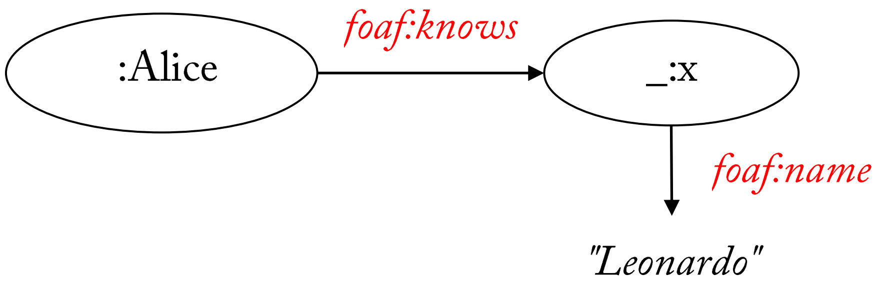
Blank nodes

- The third type of element which can appear in a triple is a blank node.
  - They represent unknown resources, which are not literals and not identified by URIs.
  - In Turtle the blank node is represented by a variable prefixed by _: , such as _:x, or as an anonymous blank node : [].

- Example (Turtle syntax)

Alice knows someone named Leonardo .

:Alice foaf:knows _:x .
_:x foaf:name "Leonardo" .



:Alice foaf:knows [foaf:name "Leonardo"] .

Instances and typing

- rdf:type models the instance relationship, or types a resource :

  ex:Bob rdf:type foaf:Person .

  The resource *Bob* is an instance of the class *Person*.

- Resources may have several classes or types :

  ex:SemanticData rdf:type ex:course .
  ex:SemanticData rdf:type ex:document .

- A class may also be an instance :

  ex:tweety rdf:type ex:robin .
  ex:robin rdf:type ex:species.

This is the first (insufficient) step to take web data towards ontologies.

Syntactic sugar

- Turtle introduces the abbreviation a to stand for rdf:type.

  :Bob a foaf:Person .

  close to the intuitive understanding *Bob (is) a Person*.

- A symbol ; separates triples with the same subject.
- A symbol , separates triples with the same predicate.

```
PREFIX ex:  <http://www.example.org>
PREFIX dc:  <http://purl.org/dc/terms/>

ex:SemanticData       dc:subject "Donnée sémantiques"@fr ,
                                 "Semantic data"@en ;
                      dc:creator  "Jean-Louis Binot".
```

The Mona Lisa example in Turtle

```
PREFIX :    <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX schema: <http://schema.org/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX wd: <http://www.wikidata.org/entity/>
:bob
        a foaf:Person ;
        foaf:knows :alice ;
        schema:birthDate "1990-07-04"^^xsd:date ;
        foaf:topic_interest wd:Q12418 .
wd:Q12418
        dcterms:title "Mona Lisa" ;
        dcterms:creator <http://dbpedia.org/resource/Leonardo_da_Vinci> .
<http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619>
        dcterms:subject wd:Q12418 .
```

Reification

- RDF provides a mechanism for reification.

- It allows to make statements about another statement seen as a resource :

  - The class rdf:Statement is used to type a resource as a statement.

  - The properties rdf:subject, rdf:predicate and rdf:object identify the subject, predicate and object of a statement.

  S rdf:subject R states that S is an instance of rdf:Statement and that the subject of S is R.

- Example : *resource item 10245 weights 2.4 Kgs. This information was created by staff member 3405.*

```
exproducts:triple12345    rdf:type        rdf:Statement .
exproducts:triple12345    rdf:subject     exproducts:item10245 .
exproducts:triple12345    rdf:predicate   exterms:weight .
exproducts:triple12345    rdf:object      "2.4"^^xsd:decimal .
exstaff:id3405            dc:creator      exproducts:triple12345 .
```

Syntaxes for expressing RDF graphs

RDF graphs can be serialized using different syntaxes :

- The Turtle family of RDF languages (N-Triples, Turtle, TriG and N-Quads).
- RDF/XML (XML syntax for RDF).
  - Initially it was the only available syntax; some people still call this syntax "RDF".
- JSON-LD (JSON for Linked Data) is a JSON-based RDF syntax.
  - Used to transform JSON documents to RDF with minimal changes, or to serialize RDF datasets in JSON.
- RDFa (RDF in HTML attributes) : used for embedding RDF in HTML and XML.

For details see the W3C manuals.

The Mona Lisa example in RDF/XML

```
<rdf:RDF    xmlns:dcterms="http://purl.org/dc/terms/"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns="http://example.org/"
            xmlns:wd="http://www.wikidata.org/entity/"
            xmlns:schema="http://schema.org/"
            xmlns:foaf="http://xmlns.com/foaf/0.1/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >
 <rdf:Description rdf:about="http://www.wikidata.org/entity/Q12418">
  <dcterms:creator rdf:resource="http://dbpedia.org/resource/Leonardo_da_Vinci"/>
  <dcterms:title>Mona Lisa</dcterms:title>
 </rdf:Description>
 <rdf:Description rdf:about="http://example.org/bob">
       <foaf:topic_interest rdf:resource="http://www.wikidata.org/entity/Q12418"/>
       <schema:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1990-07-04</schema:birthDate>
       <foaf:knows rdf:resource="http://example.org/alice"/>
       <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
 </rdf:Description>
 <rdf:Description rdf:about="http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619">
  <dcterms:subject rdf:resource="http://www.wikidata.org/entity/Q12418"/>
 </rdf:Description>
</rdf:RDF>
```

- Attributes of the rdf:RDF start tag contain namespace declarations with their abbreviations.
- rdf:Description defines sets of triples that have as subject the URI specified by *rdf:about*, following the structure:

  <*description subject*>
  <*predicate object* >
  <*predicate object* > …
  <*/end description*>

Very verbose !

RDFa (RDF in HTML attributes)

- RDFa is a serialization of RDF embedded in HTML, XML or XHTML to enrich web pages with structured semantic data.
- Why is that useful ?
  - Most of web data is in HTML; new content is generated everyday.
  - Authors of web sites traditionally do not like to generate separate RDF files.
  - Solution : add structured data to the HTML pages; let processors extract those and turn them into RDF.
  - Search engines can use these annotations to enrich search results (e.g., schema.org and Rich Snippets).
- RDFa evolved into a huge source of RDF triples on the web.
- It is a bridge between the web of documents and the web of data.

RDFa example

```
<div vocab="http://xmlns.com/foaf/0.1/" about="#me">
My name is <span property="name">John Doe</span> and my blog is called
<a rel="homepage" href="http://example.org/blog/">Understanding Semantics</a>. </div>
```

Resulting display :

My name is John Doe and my blog is called Understanding Semantics.

Resulting triples :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
<#me> foaf:name "John Doe" ;
      foaf:homepage <http://example.org/blog/> .
```

Application example : a review snippet from Google

A review snippet is a short excerpt of a review or a rating from a review website, usually an average of the combined rating scores from many reviewers. *(source https://developers.google.com/search/docs/data-types/review-snippet)*

```
<div vocab="http://schema.org/" typeof="Product">
  <img property="image" src="catcher-in-the-rye-book-cover.jpg" alt="Catcher in the Rye"/>
  <span property="name">The Catcher in the Rye</span>
  <div property="review" typeof="Review"> Review:
    <span property="reviewRating" typeof="Rating">
        <span property="ratingValue">5</span> -
    </span>
    <b>"<span property="name">A masterpiece of literature</span>" </b> by
    <span property="author" typeof="Person">
      <span property="name">John Doe</span></span>, written on
    <meta property="datePublished" content="2006-05-04">May 4, 2006
    <div property="reviewBody">I really enjoyed this book. It captures the essential challenge
people face as they try make sense of their lives and grow to adulthood.</div>
    <span property="publisher" typeof="Organization">
        <meta property="name" content="Washington Times">
    </span>
  </div>
</div>
```

The Catcher in the Rye Review: 5 - "**A masterpiece of literature**" by John Doe, written on May 4, 2006 I really enjoyed this book. It captures the essential challenge people face as they try make sense of their lives and grow to adulthood.

Triples generated from the snippet:

```
[] a ns1:Product ;
   ns1:image <catcher-in-the-rye-book-cover.jpg> ;
   ns1:name "The Catcher in the Rye" ;
   ns1:review [ a ns1:Review ;
         ns1:author [ a ns1:Person ;
               ns1:name "John Doe" ] ;
         ns1:datePublished "2006-05-04" ;
         ns1:name "A masterpiece of literature" ;
         ns1:publisher [ a ns1:Organization ;
               ns1:name "Washington Times" ] ;
         ns1:reviewBody "I really enjoyed this book.
It captures the essential challenge people face as
they try make sense of their lives and grow to
adulthood." ;
         ns1:reviewRating [ a ns1:Rating ;
               ns1:ratingValue "5" ] ] .
```

Formal semantics of RDF

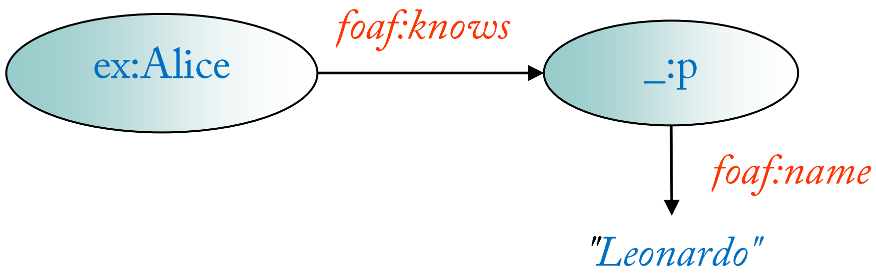
Why does RDF need semantics?

- Semantic networks by themselves do not have clear semantics (cf. chapter 1).
- RDF is the basis for more advanced ontology languages such as RDFS and OWL.
  - These languages aim to model knowledge and support inferences; they need formal semantics.
- Semantic web query engines using SPARQL operate on RDF documents.
  - Tools were incompatible : same RDF documents, same SPARQL queries, different results !
- A formal semantics was defined for RDF (W3C document [*RDF11-MT*]).
  - The approach of reference is to define directly a model-theoretic semantics for RDF.
  - For the sake of simplicity, we will introduce the main points by referring to first order logic.

Semantics of RDF graphs

- Syntax : sentences of the language :
  - Any statement or triple of the form *Subject Predicate Object* is a basic sentence of the RDF language(*).
  - Complex sentences of the RDF language are finite sets of triples, forming an RDF graph.

- Semantics : basic rules :
  - An RDF statement can be interpreted as a FOL formula : *Predicate(Subject, Object)* .
  - A set of triples is interpreted as a conjunction. An RDF graph is *True* exactly when all the triples in it are *True*.
  - URIs used in subject, predicate, and object are global in scope, naming the same thing each time they are used.
  - A blank node corresponds to an existentially quantified variable.



∃p (knows(ex:Alice, p) ∧ name(p, "Leonardo"))

(prefixes omitted for the sake of readability)

*: from a syntactic point of view several serializations are possible.

Semantics of RDF graphs ./.

- Using our usual model-theoretic terminology, we will say that :
  - An interpretation $\mathcal{I}$ satisfies a statement or triple of the form s p o if $p^{\mathcal{I}}(s^{\mathcal{I}}, o^{\mathcal{I}})$ is *True*.
  - An interpretation $\mathcal{I}$ satisfies a graph G if every triple included in G is *True* under $\mathcal{I}$.
  - A graph G is satisfiable if at least one interpretation satisfies it.
  - A graph G entails a graph E, noted $G \models E$, when every interpretation which satisfies G also satisfies E.
  - If two graphs E and F entail each other, they are logically equivalent.

Note : To be complete, the definitions of this section concern the concepts of **simple interpretation** and **simple entailment**. RDF is intended to be used as a basis for other languages (RDFS, OWL) and more complex entailment regimes will occur for them.

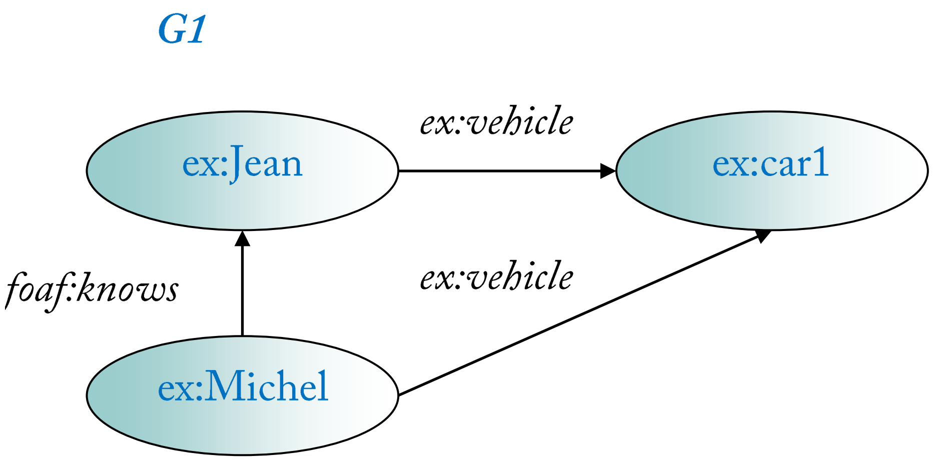
The interpolation lemma

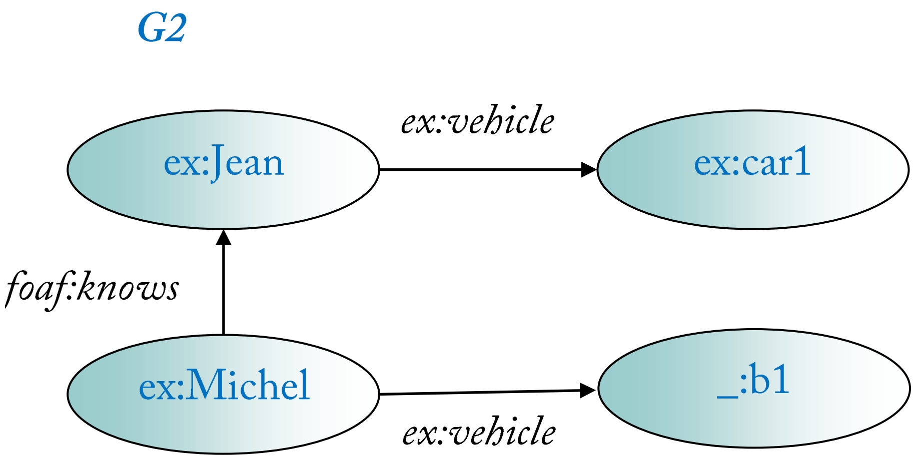
The interpolation lemma allows to check RDF graph entailment on a syntactic basis.

- Subgraph : a subgraph of an RDF graph is a subset of the triples in the graph.
  - A graph entails all its subgraphs.
- Graph instance :

  if M is a functional mapping from a set of blank nodes to another set of literals, blank nodes and URIs,
  - Any graph obtained from a graph G by replacing a subset N of the blank nodes in G by M(N) is an instance of G.
  - A graph is entailed by any of its instances.
- Interpolation lemma :
  - If G1 and G2 are two RDF graphs, G1 ⊨ G2 iff there is a subgraph of G1 which is an instance of G2.
  - It follows directly from the two previous points.

Interpolation lemma: example





- There exists a subgraph of G1 which is an instance of G2. Hence G1 ⊨ G2
  - Using the mapping M(_:b1)=ex:car1.
- There is no subgraph of G2 being an instance of G1. Hence G2 ⊭ G1
- Simple graph entailment is decidable and NP-complete in general.
  - Complexity reduces to polynomial order when G2 contains no blank node .

(example from Apprentissage symbolique et web sémantique, UPMC)

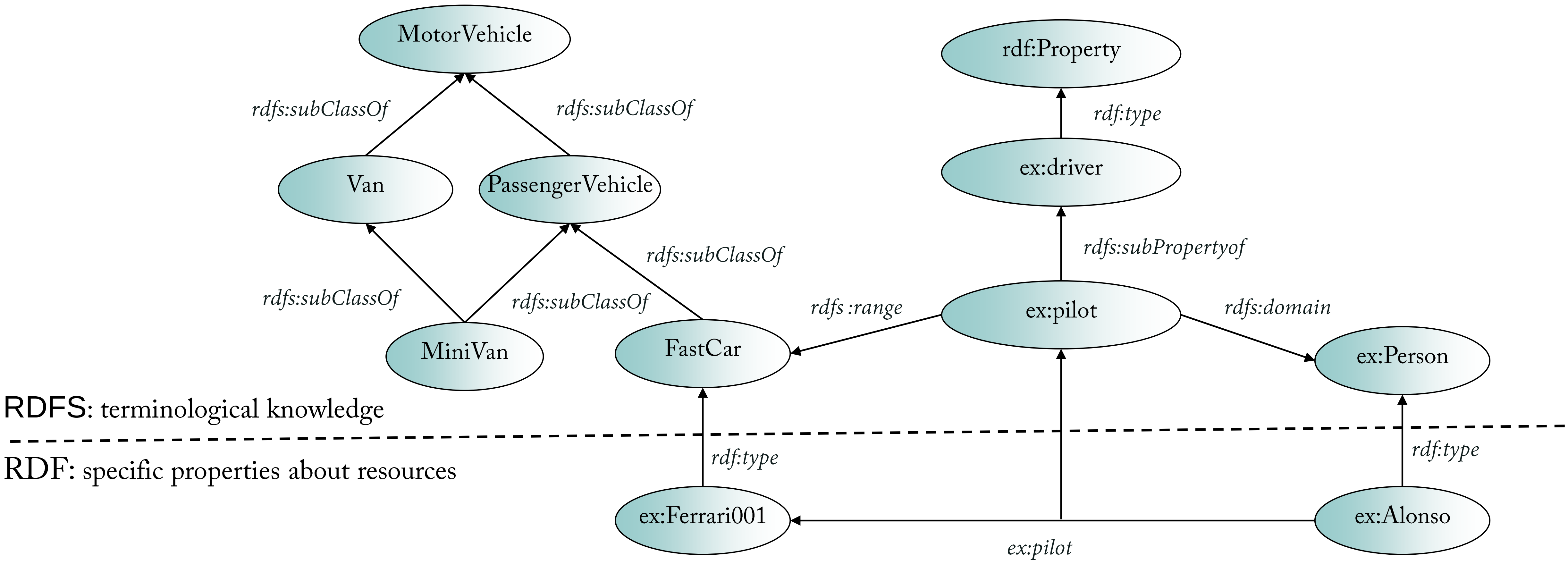
Agenda

Motivation for RDFS

- RDF is a universal language annotating web resources with specific properties.
- It does not say anything more about the meaning of the resources or annotations.
  - And, unlike i.a. Dublin Core, RDF does not predefine any application-specific terminology.
- RDF as such does not allow us to build ontologies.
- RDF Schema (RDFS) is a first step in that direction (W3C' *one step at a time* strategy) :

  RDFS allows to express **simple** ontologies by adding a few constructs to RDF.

Example of a simple RDFS ontology



RDFS basic ideas

- RDFS provides syntactic constructs allowing to :
  - Specify concepts (classes), class hierarchies and instances.
  - Specify properties (roles) and property hierarchies.
  - Specify which classes and properties should be used together.
  - Support inheritance.
- RDFS acts as a typing system for RDF resources.
- RDFS statements are RDF triples. RDFS ontologies are valid RDF graphs.
  - RDFS constructs are provided as an RDF vocabulary; its namespaces and usual prefix are :

    rdfs: <http://www.w3.org/2000/01/rdf-schema#>

RDFS main constructs

Construct (construct type)	Syntactic form	Description
Class (a class)	**C** rdf:type rdfs:Class	**C** (a resource) is an RDFS class
Property (a class)	**P** rdf:type rdf:Property	**P** (a resource) is an RDF property
type (a property)	**I** rdf:type **C**	**I** (a resource) is an instance of **C** (a class)
subClassOf (a property)	**C1** rdfs:subClassOf **C2**	**C1** (a class) is a subclass of **C2** (a class)
subPropertyOf (a property)	**P1** rdfs:subPropertyOf **P2**	**P1** (a property) is a sub-property of **P2** (a property)
domain (a property)	**P** rdfs:domain **C**	domain of **P** (a property) is **C** (a class)
range (a property)	**P** rdfs:range **C**	range of **P** (a property) is **C** (a class)

Defining classes and instances

Example : an organization wants to provide information about motor vehicles.

- The following RDF triples define classes :

  ex:MotorVehicle rdf:type rdfs:Class .

  ex:Van rdf:type rdfs:Class .

  ex:Truck rdf:type rdfs:Class .

- The following RDF triple defines companyCar as an instance of MotorVehicle :

  exthings:companyCar rdf:type ex:MotorVehicle .

Notes :
1) we use again Turtle syntax.
2) by convention class names have an initial uppercase letter; property and instance names have an initial lowercase.

Defining classes and instances ./.

- A class is any resource having an rdf:type whose value is the resource rdfs:Class.

  The statement C rdf:type rdfs:Class . declares C as a class.

- An instance is also declared by using the rdf:type property.

  The statement I rdf:type C . declares that I (a resource) is an instance of C (a class).

  - The same statement is also interpreted as an implicit declaration that C is an instance of rdfs:Class.

    Class declarations are not mandatory; but they are useful for ontology documentation.

  - The declaration C rdf:type rdfs:Class . is also saying that C is an instance of rdfs:Class.

    Any class is an instance of rdfs:Class.

- ⚠ The resource rdfs:Class itself is an instance of rdfs:Class :
  => a class may be a member of its own extension (an instance of itself).

Subsumption

- Subsumption is expressed by the predefined property rdfs:subClassOf.

  The statement C1 rdfs:subClassOf C2 . declares that class C2 subsumes C1.

- rdfs:subClassOf is transitive : a class is a subclass of all its super classes.
  This is captured by an RDFS semantic inference rule, or entailment pattern:

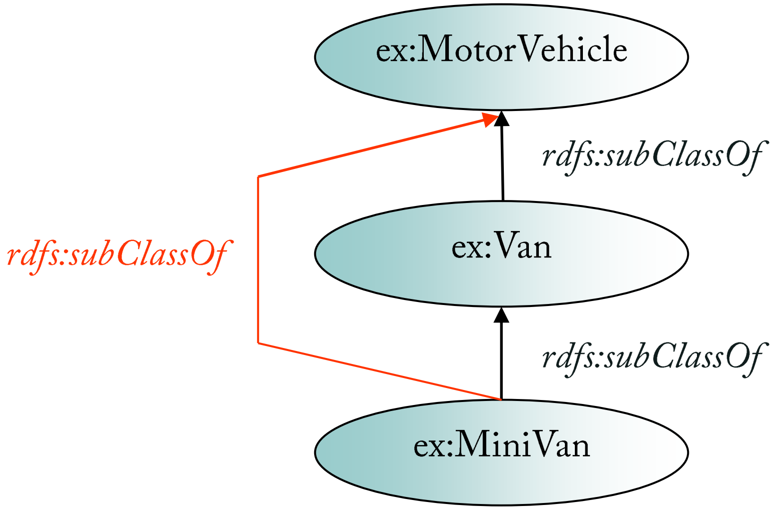
C rdfs:subClassOf D . ∧ D rdfs:subClassOf E . entails C rdfs:subClassOf E .

- Example : from :

  ex:Van rdfs:subClassOf ex:MotorVehicle .
  ex:MiniVan rdfs:subClassOf ex:Van .

  we can infer :

  ex:MiniVan rdfs:subClassOf ex:MotorVehicle .



Class hierarchy and instances

- An instance of a class is also an instance of its super classes.
  Semantic inference rule :

C rdfs:subClassOf D . ∧ I rdf:type C . entails I rdf:type D .

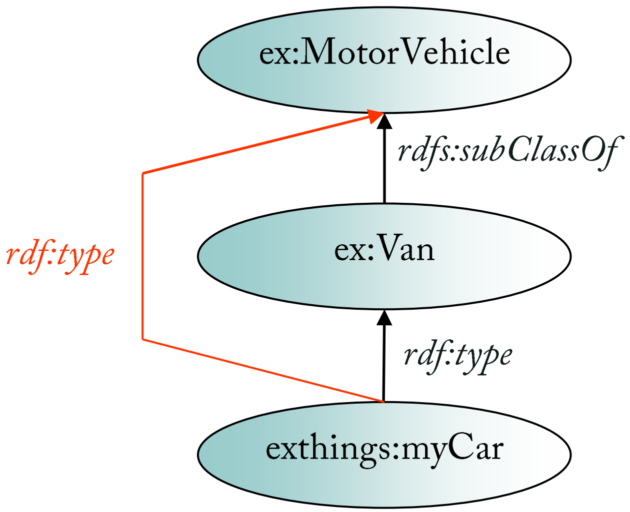
- Example: from the following statements :

  ex:Van rdfs:subClassOf ex:MotorVehicle .

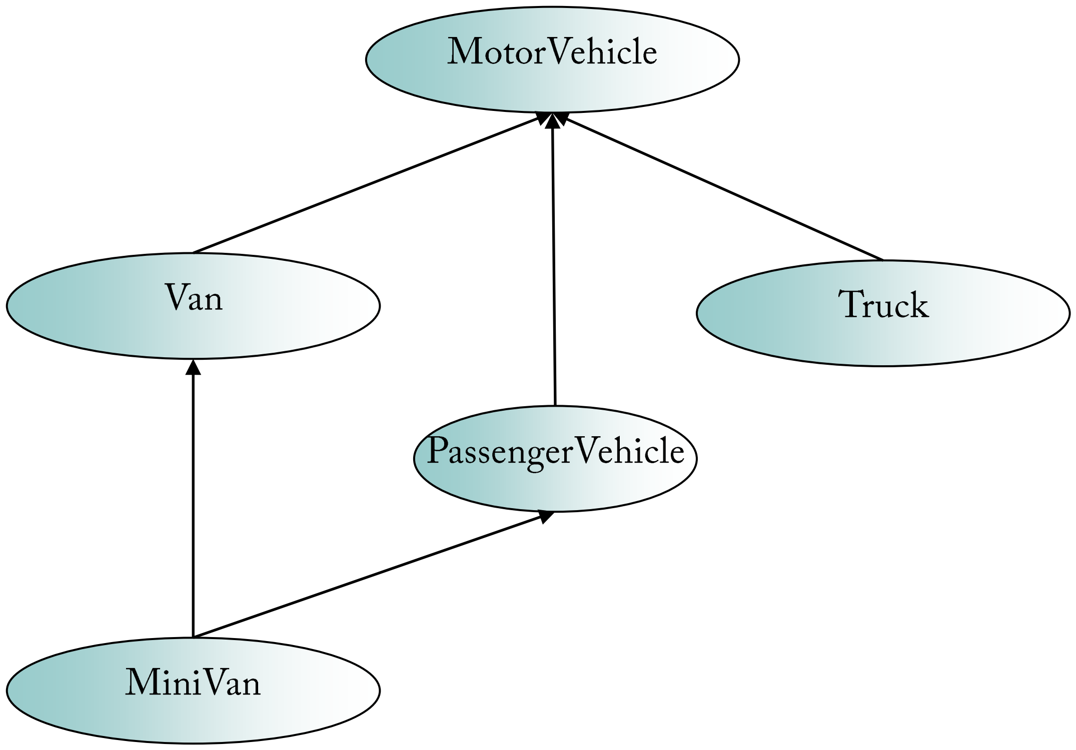
  exthings:myCar rdf:type ex:Van .

  RDFS allows us to infer :

  exthings:myCar rdf:type ex:MotorVehicle .



Example of hierarchy with corresponding triples



(Source: RDF primer 2004)

ex:MotorVehicle	*rdf:type*	*rdfs:Class .*
ex:PassengerVehicle	*rdf:type*	*rdfs:Class .*
ex:Van	*rdf:type*	*rdfs:Class .*
ex:Truck	*rdf:type*	*rdfs:Class .*
ex:MiniVan	*rdf:type*	*rdfs:Class .*
ex:PassengerVehicle	*rdfs:subClassOf*	*ex:MotorVehicle .*
ex:Van	*rdfs:subClassOf*	*ex:MotorVehicle .*
ex:Truck	*rdfs:subClassOf*	*ex:MotorVehicle .*
ex:MiniVan	*rdfs:subClassOf*	*ex:Van .*
ex:MiniVan	*rdfs:subClassOf*	*ex:PassengerVehicle .*

Strict multiple inheritance is allowed!

Defining Properties

- All properties in RDF are described as instances of class rdf:Property.

  The statement *P rdf:type rdf:Property .* declares that P is a property :

  Examples :

  exterms:weightInKg rdf:type rdf:Property .
  ex:hasAuthor rdf:type rdf:Property .

- Properties can be used in any RDF triple :

  ex:TheNameOfTheRose ex:hasAuthor ex:UmbertoEco .

- Declaring a property is optional. Semantic inference rule :

subject property object . entails property rdf:type rdf:Property .

Range and domain of properties

- The range of a property (the set of values it can take) is expressed by rdfs:range.

  The statement P rdfs:range C . indicates that the values of property P are instances of class C.

  Example :

ex:Person	*rdf:type*	*rdfs:Class .*
ex:hasAuthor	*rdf:type*	*rdf:Property .*
ex:hasAuthor	*rdfs:range*	*ex:Person .*

- The domain of a property (the set of values to which it can be applied) is expressed by rdfs:domain.

  The statement P rdfs:domain C . indicates that property P applies to values from class C.

  Example (added to the above triples) :

  ex:hasAuthor rdfs:domain ex:Book .

Range and domain inference rules

- If a property P has a range property, any value of P must be a member of the range class :

P rdfs:range C . ∧ X P Y . entails Y rdf:type C .

- If a property P has a domain property, any resource to which P is applied must be a member of the domain class :

P rdfs:domain C . ∧ X P Y . entails X rdf:type C .

- Example : from the following statements :

ex:Book	*rdf:type*	*rdfs:Class .*
ex:Person	*rdf:type*	*rdfs:Class .*
ex:hasAuthor	*rdf:type*	*rdf:Property .*
ex:hasAuthor	*rdfs:domain*	*ex:Book .*
ex:hasAuthor	*rdfs:range*	*ex:Person .*
ex:MobyDick	*ex:hasAuthor*	*ex:Stevenson .*

we can deduce :

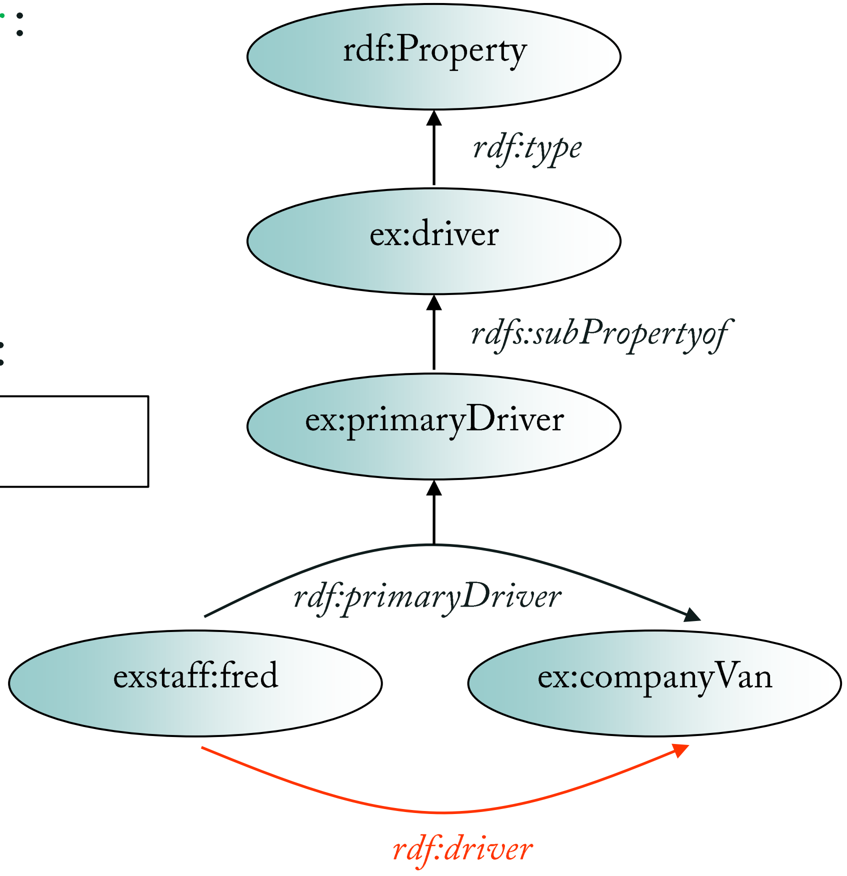
ex:Stevenson rdf:type ex:Person .
ex:MobyDick rdf:type ex:Book .

Specialization of properties

- RDF Schema allows to specialize properties, using rdfs:subPropertyOf.

  Example : *ex:primaryDriver* is a specialization of *ex:driver* :

ex:driver	*rdf:type*	*rdf:Property .*
ex:primaryDriver	*rdf:type*	*rdf:Property .*
ex:primaryDriver	*rdfs:subPropertyOf*	*ex:driver .*



- Semantic inference rule exploiting the property hierarchy :

> P rdfs:subPropertyOf Q . ∧ X P Y . entails X Q Y .

Example :

from	*exstaff:fred ex:primaryDriver ex:companyVan .*
we can deduce	*exstaff:fred ex:driver ex:companyVan .*

Complex cases for range and domain of properties

⚠ The rdfs:range property can be applied to itself.

rdfs:range rdfs:range rdfs:Class .

Any resource which is the value of rdfs:range property is an instance of rdfs:Class.

⚠ The rdfs:domain property can be applied to itself.

rdfs:domain rdfs:domain rdf:Property .

Any resource with an rdfs:domain property is an instance of rdf:Property.

Such possibilities are expensive : they have an important impact on the complexity of the language, as we will see later.

Multiple range or domain statements

- Two range statements may apply to the same property :

  P rdfs:range C1 .
  P rdfs:range C2 .

  The values of property P are instances of both C1 and C2.

- Example : from

  ex:hasMother rdfs:range ex:Person .
  ex:hasMother rdfs:range ex:Female .
  exstaff:frank ex:hasMother exstaff:frances .

  we can infer that *exstaff:frances* is an instance of both *ex:Female* and *ex:Person*.

- A similar conclusion applies for two domain statements.

Containers

- RDF containers are resources used to represent collections (lists, sets…).
- Three classes are available with the same semantics, but a different intended use:
  - rdf:Bag : used to indicate that the content of the container is an unordered set.
  - rdf:Seq : used to indicate that the ordering of the elements in the container is important.
  - rdf:Alt : used to indicate that a typical processing will only select one element of the container.

  All three are subclasses of rdfs:Container.
- Access to specific items in a container :
  - Properties rdf:_1, rdf:_2, rdf:_3… are used to state that a resource is a member of a container.
  - A statement of the form C rdf:_nnn O states that O is member *nnn* of container C.

RDFS Semantics

The formal semantics of RDF and RDFS is defined in the W3C document [*RDF11-MT*].

We will again summarize the key points by referring to first order logic.

- RDFS statements can be interpreted by FOL formulas as indicated below :

RDF/RDFS Statement	FOL Formula
<s P o>	P(s, o)
i rdf:type C	C(i)
C1 rdfs:subclass C2	$\forall x\ (C1(x) \rightarrow C2(x))$
P1 rdfs:subPropertyof P2	$\forall x\ \forall y\ (P1(x, y) \rightarrow P2(x, y))$
P rdfs:domain C	$\forall x\ \forall y\ (P(x, y) \rightarrow C(x))$
P rdfs:range D	$\forall x\ \forall y\ (P(x, y) \rightarrow D(y))$

RDFS Semantics

- Entailment patterns :
  - Given a set of RDF(S) triples accepted as *True*, one can deduce that other RDF(S) triples must be *True*.
  - These entailment patterns, or inference rules, can be used by a reasoner.
  - They can also be used operationally to generate new triples until it is no longer possible to do so.
    At this stage, the RDFS graph will be called saturated.

    The saturated graph does not change entailment and is semantically equivalent to the original one.
    It allows to decouple the execution of queries against the graph from the computation of entailment.

- RDF and RDFS entailment patterns have been illustrated in previous slides.
  A complete list is given in annex and be found in the following places :
  - https://www.w3.org/TR/rdf11-mt/#patterns-of-rdf-entailment-informative
  - https://www.w3.org/TR/rdf11-mt/#patterns-of-rdfs-entailment-informative

Revisiting the example from chapter 1

:country a rdf:class .
:city a rdf:class .
:locatedIn a :rdf:property .

:capitalOf rdfs:subPropertyOf :locatedIn .
:capitalOf rdfs:domain :city .
:capitalOf rdfs:range :country .

“Madrid is the capital of Spain”

:Madrid :capitalOf :Spain .

Madrid is a city.
Spain is a country.
Madrid is located in Spain.
Barcelona is not the capital of Spain.
Madrid is not the capital of France.
Madrid is not a country.
...

Madrid is a city.

:Madrid :capitalOf :Spain .
\+ :capitalOf rdfs:domain :city .

=> :Madrid a :city .

Spain is a country.

:Madrid :capitalOf :Spain .
\+ :capitalOf rdfs:range :country .

=> :Spain a :country .

Madrid is located in Spain.

:Madrid :capitalOf :Spain .
\+ :capitalOf rdfs:subPropertyOf :locatedIn .

=> :Madrid :locatedIn :Spain .

(example after Paulheim, Semantic Web Technologies)

Revisiting the example from chapter 1

:locatedIn a :rdf:property .
:country a rdf:class .
:city a rdf:class .
:capitalOf rdfs:subPropertyOf :locatedIn .
:capitalOf rdfs:domain :city .
:capitalOf rdfs:range :country .

"Madrid is the capital of Spain"

:Madrid :capitalOf :Spain .

Madrid is a city.
Spain is a country.
Madrid is located in Spain.
Barcelona is not the capital of Spain.
Madrid is not the capital of France.
Madrid is not a country.
...

✓ Madrid is a city.

✓ Spain is a country.

✓ Madrid is located in Spain.

✗ Barcelona is not the capital of Spain.
Every country has exactly one capital (cardinality restrictions).

✗ Madrid is not the capital of France.
A city is the capital of at most one country (functional properties).

✗ Madrid is not a country.
A city cannot be a country at the same time (disjoint classes).

❑ We need a more expressive language !

(example after Paulheim, Semantic Web Technologies)

Summary

- RDF allows to represent web resources, identified by URIs. Its data structure is based on triples. A set of triples builds the equivalent of a semantic network.
- RDF has a clear model-theoretic semantics definition. It however lacks any specific constructs to represent ontologies.
- RDFS extends RDF with basic constructs such as classes, subclasses, and properties, allowing to start building simple ontologies.
- RDFS has a number of entailment patterns which can be used to make inferences by producing new triples.
- RDFS also has limitations in terms of knowledge representation.
  We need a more expressive language to build complex ontologies.

Annex : RDF/RDFS entailment patterns

	If S contains:	then S RDF(S)_entails :
rdfD1	a p "v"^^d .	a p _:b . _b rdf:type d .
rdfD2	a p b .	p rdf:type rdf:Property .
rdfs1	any IRI a in D	a rdf:type rdfs:Datatype .
rdfs2	p rdfs:domain x . a p b .	a rdf:type x .
rdfs3	p rdfs:range x . a p b .	b rdf:type x .
rdfs4a	a p b .	a rdf:type rdf:Resource .
rdfs4b	a p b .	b rdf:type rdf:Resource .
rdfs5	p rdfs:subPropertyOf q . q rdfs:subPropertyOf r .	p rdfs:subPropertyOf r .
rdfs6	a rdf:type rdf:Property .	a rdfs:subPropertyOf a .
rdfs7	rdfs:subPropertyOf q . a p b.	a q b .
rdfs8	c rdf:type rdfs:Class .	c rdfs:subClassOf rdf:Resource .
rdfs9	c rdfs:subClassOf d . a rdf:type c.	a rdf:type d .
rdfs10	c rdf:type rdfs:Class .	c rdfs:subClassOf c .
rdfs11	c rdfs:subClassOf d . d rdfs:subClassOf e .	c rdfs:subClassOf e .
rdfs12	rdf:type rdfs:ContainerMembershipProperty .	p rdfs:subPropertyOf rdfs:member .
rdfs13	d rdf:type rdfs:Datatype .	d rdfs:subClassOf rdfs:Literal .

References

- *[Abiteboul et al. 2011]: Abiteboul S., Manolescu I, Rigaux P., Rousset M-C. and Senellart P., Web Data Management, Cambridge University Press, 2011.*
- *[Antoniou and Van Harmelen 2004]: Antoniou G. and Van Harmelen F.,* A semantic web primer, *MIT Press, 2004.*
  Information available at http://www.ics.forth.gr/isl/swprimer/.

THANK YOU