RDF and RDF Schema

Oscar Corcho, Raúl García Castro, Oscar Muñoz-García
{ocorcho,rgarcia}@fi.upm.es, omunoz@delicias.dia.fi.upm.es
http://www.oeg-upm.net/

Ontological Engineering Group
Laboratorio de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn,
28660 Boadilla del Monte, Madrid, Spain

Main References

Gómez-Pérez, A.; Fernández-López, M.; Corcho, O. **Ontological Engineering.** *Springer Verlag. 2003*

Capítulo 4: Ontology languages

Brickley D, Guha RV (2004) *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation.
http://www.w3.org/TR/PR-rdf-schema
Lassila O, Swick R (1999) *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation.
http://www.w3.org/TR/REC-rdf-syntax/
Prud'hommeaux E, Seaborne A (2008) *SPARQL Query Language for RDF*. W3C Recommendation.
http://www.w3.org/TR/rdf-sparql-query/

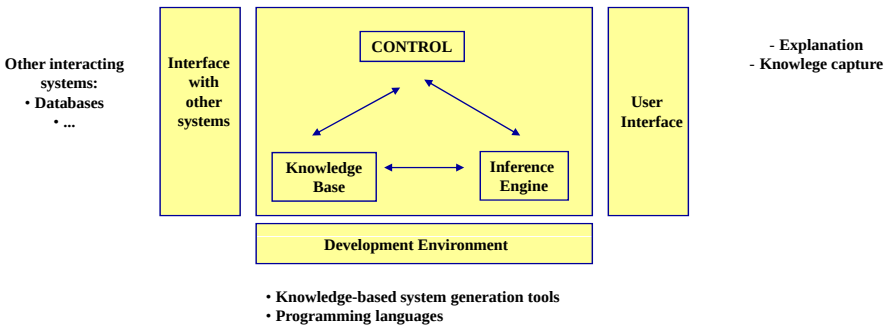
Jena web site: *http://jena.sourceforge.net/*
Jena API: *http://jena.sourceforge.net/tutorial/RDF_API/*
Jena tutorials: *http://www.ibm.com/developerworks/xml/library/j-jena/index.html*
http://www.xml.com/pub/a/2001/05/23/jena.html

SPARQL validator: *http://www.sparql.org/validator.html*
SPARQL implementations: *http://esw.w3.org/topic/SparqlImplementations*
SPARQL tutorials: *http://jena.sourceforge.net/ARQ/Tutorial/*
http://www.w3.org/2004/Talks/17Dec-sparql/intro/all.html
http://www.cs.man.ac.uk/~bparsia/2006/row-tutorial/

Table of Contents

1. An introduction to knowledge representation formalisms
2. Resource Description Framework (RDF)
   - 2.1. RDF primitives
   - 2.2. Reasoning with RDF
3. RDF Schema
   - 3.1 RDF Schema primitives
   - 3.2 Reasoning with RDFS
4. RDF(S) management APIs
5. RDF(S) query languages: SPARQL

Common Architecture of a Knowledge-based System



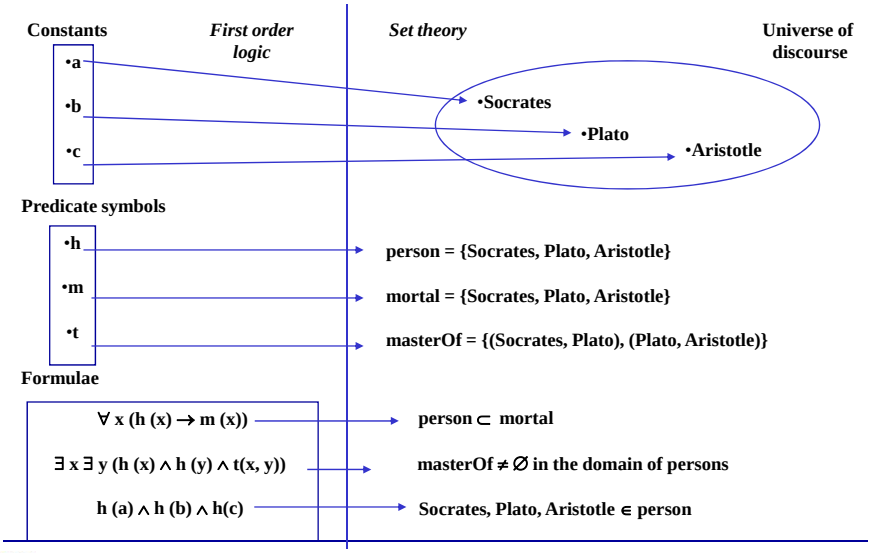
Knowledge Representation Formalisms. A Summary

- **Knowledge representation**
  - To store knowledge so that programs can process it and achieve the verisimilitude of human intelligence
- **Knowledge representation formalisms/techniques**
  - Originated from theories of human information processing.
  - Since knowledge is used to achieve intelligent behavior, the fundamental goal of knowledge representation is to represent knowledge in a manner as to facilitate inferencing i.e. drawing conclusions from knowledge.
  - Some examples are:
    - First order logic
    - Semantic networks and conceptual maps
    - Frames
    - Description logic
    - Production rules
    - Fuzzy logic
    - Bayesian networks
    - Etc.

These are the ones that we will analyse

First order logic. Basic elements

We can establish mappings between logical symbols and domain objects (universe of discourse)



First order logic. Formalisation

- **We have a robot that delivers boxes to offices. We know:**
  - Boxes in room 27 are smaller than those in room 28.
  - All boxes in the same room are of the same size.
  - In a given moment in time, we know:
    - i) Box A is inside room 27 or 28 (we do not know which one).
    - ii) Box B is inside room 27.
    - iii) Box B is not smaller than box A.
  - We want to test whether box A is in room 27.

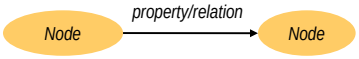
First order logic. Formalisation. Solution

- **We have a robot that delivers boxes to offices. We know:**
  - Boxes in room 27 are smaller than those in room 28.
    $\forall x \forall y (box(x) \wedge inside(x,h27) \wedge box(y) \wedge inside(y,h28) \rightarrow smallerThan(x,y))$
  - All boxes in the same room are of the same size.
    $\forall x \forall y \forall h (box(x) \wedge box(y) \wedge room(h) \wedge room(x,h) \wedge inside(y,h) \rightarrow sameSizeAs(x,y))$
  - In a given moment in time, we know :
    - i) Box A is inside room 27 or 28 (we do not know which one).
      $box(a) \wedge room(h27) \wedge room(h28) \wedge (inside(a,h27) \vee inside(a,h28))$
    - ii) Box B is inside room 27.
      $box(b) \wedge inside(b,h27)$
    - iii) Box B is not smaller than box A.
      $\neg smallerThan(b,a)$
  - We want to test whether box A is in room 27.
    ¿inside(a,h27)?

Semantic Network. Basic elements

- **Nodes**
  - They represent entities or concepts, or values

- **Edges**
  - They represent properties or relations



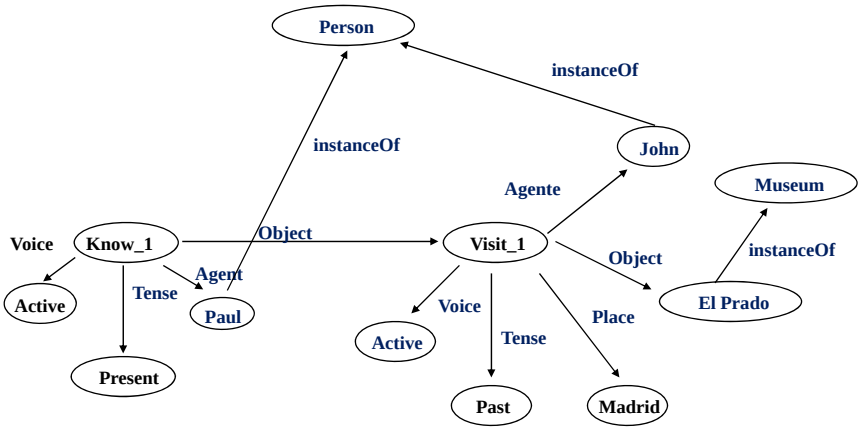
- **The semantics (mapping to the real world) depends on the tags used for nodes and edges**
- **There is no predefined KR vocabulary**
  - Although sometimes there are *structural* edges

Semantic networks. Example

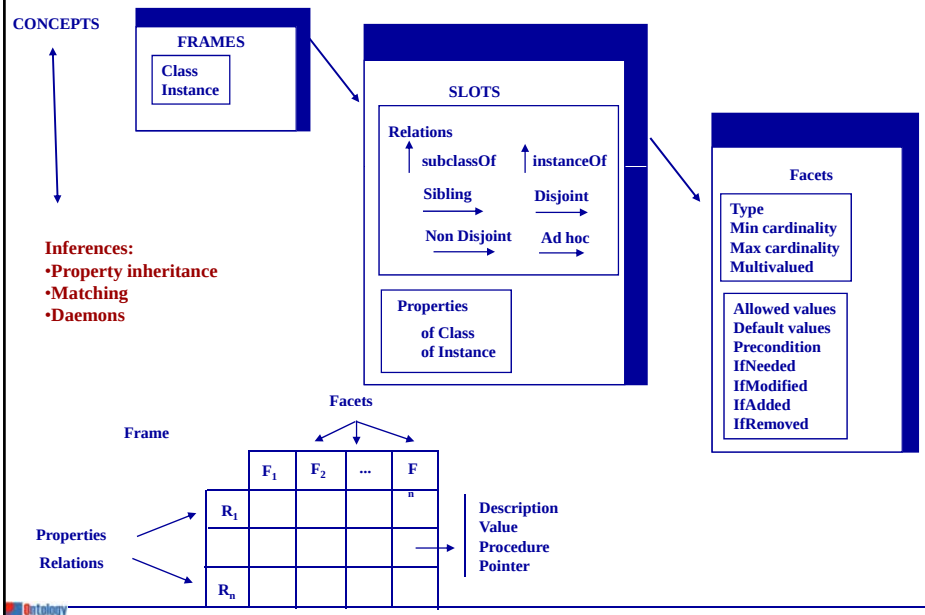
- **Paul and John are persons**
- **El Prado is a museum**
- **Paul knows that John visited El Prado in Madrid**

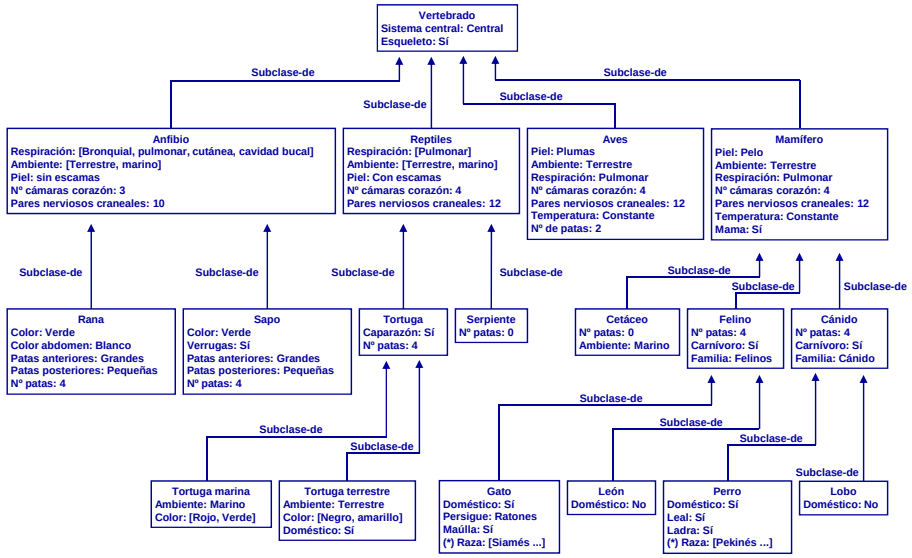
Semantic networks. Example. Solution

- Paul and John are persons
- El Prado is a museum
- Paul knows that John visited El Prado in Madrid



Frames. Basic elements



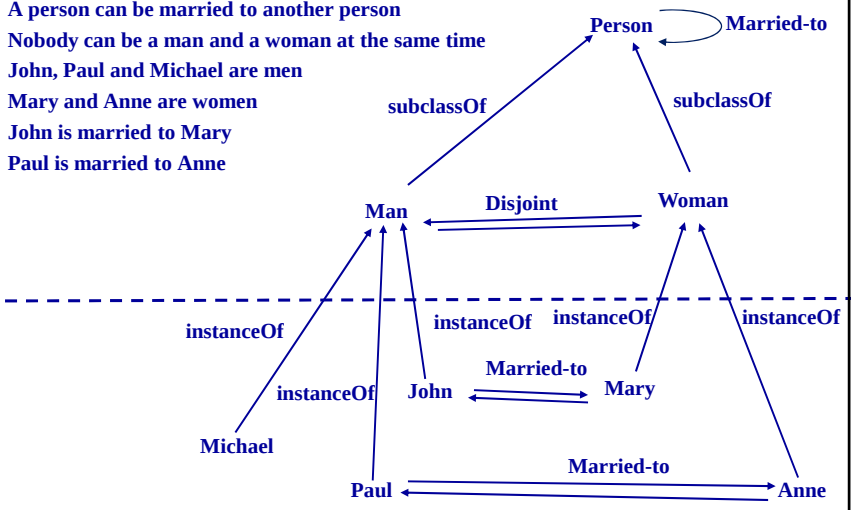


Frames. Example

- Men and women are persons
- A person can be married to another person
- Nobody can be a man and a woman at the same time
- John, Paul and Michael are men
- Mary and Anne are women
- John is married to Mary
- Paul is married to Anne

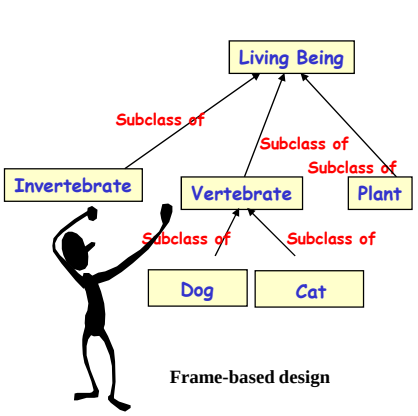
Frames. Example

- Men and women are persons
- A person can be married to another person
- Nobody can be a man and a woman at the same time
- John, Paul and Michael are men
- Mary and Anne are women
- John is married to Mary
- Paul is married to Anne

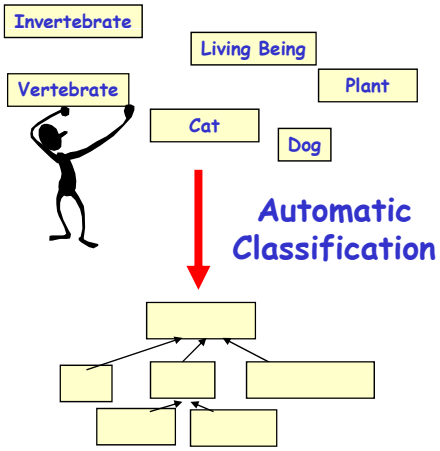


Description Logics. Basic elements

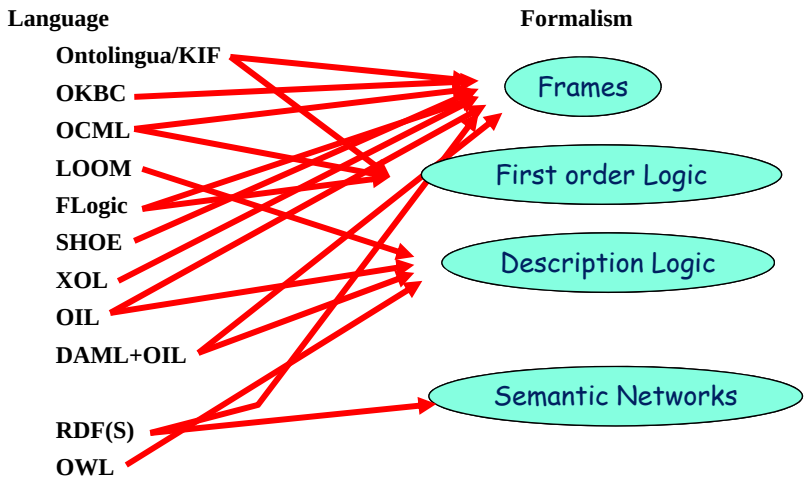
- A subset of first order logic with good reasoning properties
- Automatic classification



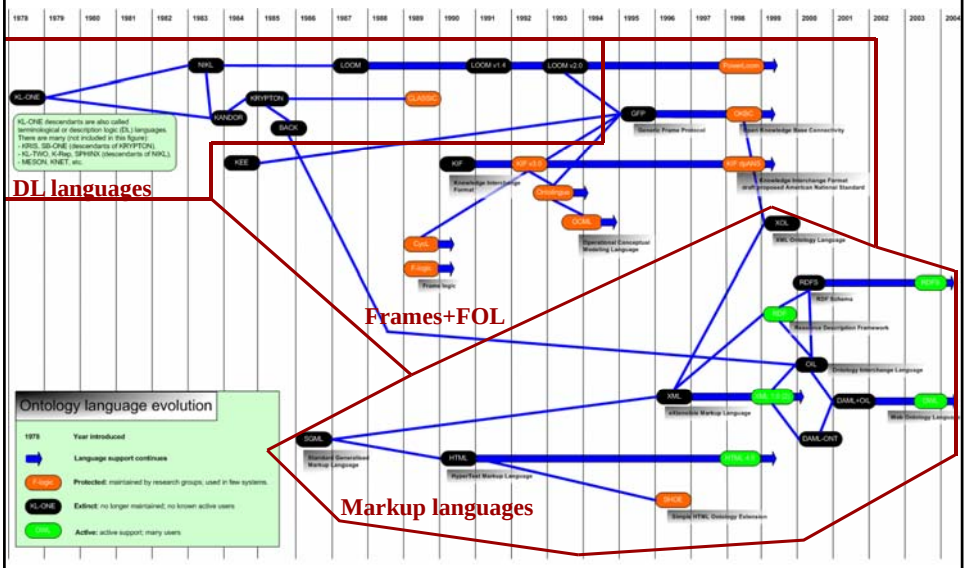
Frame-based design



KR Formalisms



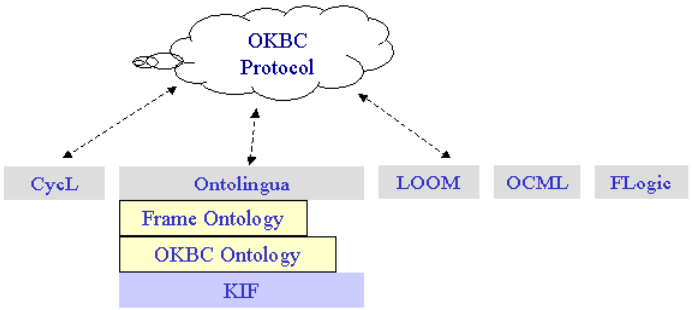
Ontology language evolution



Ontology Languages (I)

Traditional ontology languages

- **Ontolingua/KIF**
- **OKBC**
- **OCML**
- **LOOM**
- **FLogic**



Ontology Languages (II)

Ontology markup languages

- **Standards & Recommendations of W3C**
  - **XML**
  - **RDF(S)**
- **Ontology specification languages**
  - **SHOE**
  - **OIL**
  - **DAML+OIL**
  - **OWL**
  - **XOL**

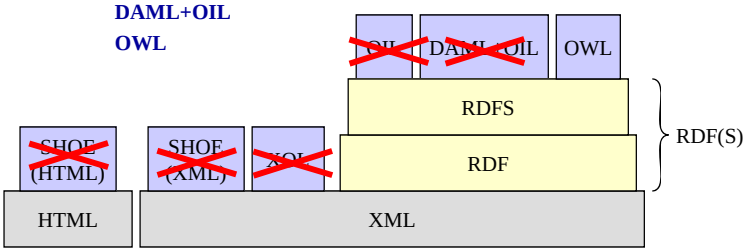
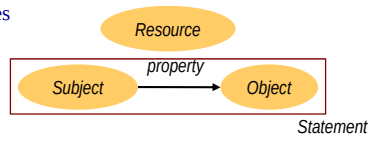


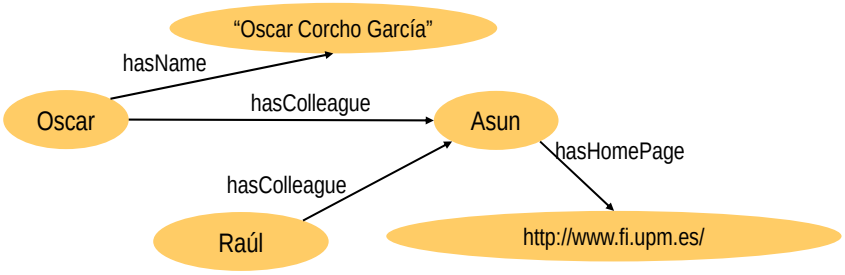
Table of Contents

1. An introduction to knowledge representation formalisms
2. Resource Description Framework (RDF)
   - 2.1. RDF primitives
   - 2.2. Reasoning with RDF
3. RDF Schema
   - 3.1 RDF Schema primitives
   - 3.2 Reasoning with RDFS
4. RDF(S) management APIs
5. RDF(S) query languages: SPARQL

RDF: Resource Description Framework

- **W3C recommendation**
- **RDF is a basic KR language, based on semantic networks**
  - Useful to represent metadata and describe any type of information in a machine-accessible way (aka data model)
  - Resources are described in terms of properties and property values using RDF statements.
  - Statements are represented as triples, consisting of a subject, predicate and object. [S, P, O]



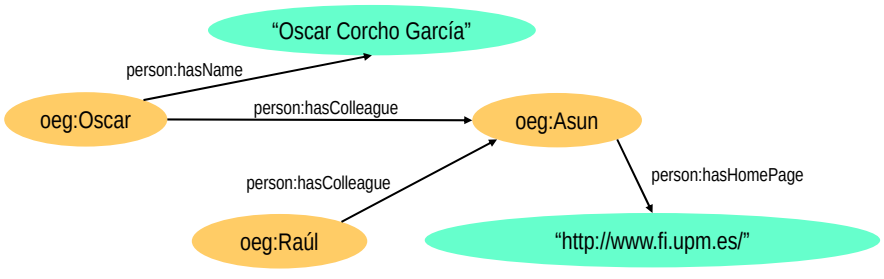


RDF (and other W3C Recommendations) and URIs

- A URI (Unique Resource Identifiers) is a Web identifier
  - e.g. http://www.oeg-upm.net/ontologies/people#Oscar
  - URI ≠ URL
    - If we open a Web browser and point to that URI, the corresponding object will not be necessarily downloaded or shown
    - If URLs work for **locating** uniquely (with no collisions) a Web page/resource, why not using the same approach for **identifying** Web resources?
  - Other valid URIs could be
    - ftp://www.oeg-upm.net/ontologies/people#Oscar
    - persons://www.oeg-upm.net/ontologies/people#Oscar
    - ...
- URIs allow identifying
  - Individuals: http://www.oeg-upm.net/ontologies/people#Oscar
  - Kinds of things: http://www.ontologies.org/ontologies/people#Person
  - Properties of those things: http://www.ontologies.org/ontologies/people#hasColleague
- **Beware!! This is changing in the context of Linked Data!!**

RDF (and other W3C Recommendations) and URIs

- For practical purposes, especially if handwritten, URIs are shortened using XML namespaces
  - xmlns:oeg="http://www.oeg-upm.net/ontologies/people#"
  - oeg:Oscar is equivalent to http://www.oeg-upm.net/ontologies/people#Oscar



RDF Serialisations

- **Normative**
  - RDF/XML (www.w3.org/TR/rdf-syntax-grammar/)
- **Alternative (for human consumption)**
  - N3 (http://www.w3.org/DesignIssues/Notation3.html)
  - Turtle (http://www.dajobe.org/2004/01/turtle/)
  - TriX (http://www.w3.org/2004/03/trix/)
  - …

Important note: the order of RDF statements in a serialisation does not affect the behaviour of a parser/application

RDF Serialisations. RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:person="http://www.ontologies.org/ontologies/people#"
  xmlns="http://www.oeg-upm.net/ontologies/people#"
  xml:base="http://www.oeg-upm.net/ontologies/people">

  <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasHomePage"/>
  <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasColleague"/>
  <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasName"/>

  <rdf:Description rdf:about="#Raúl"/>
  <rdf:Description rdf:about="#Asun">
    <person:hasColleague rdf:resource="#Raúl"/>
    <person:hasHomePage>http://www.fi.upm.es</person:hasHomePage>
  </rdf:Description>
  <rdf:Description rdf:about="#Oscar">
    <person:hasColleague rdf:resource="#Asun"/>
    <person:hasName>Oscar Corcho García</person:hasName>
  </rdf:Description>

</rdf:RDF>
```

RDF Serialisations. N3

```
@base  <http://www.oeg-upm.net/ontologies/people >
@prefix person: <http://www.ontologies.org/ontologies/people#>
:Asun    person:hasColleague :Raúl ;
         person:hasHomePage “http://www.fi.upm.es/”.
:Oscar   person:hasColleague :Asun ;
         person:hasName “Oscar Corcho García”.
```


Exercise

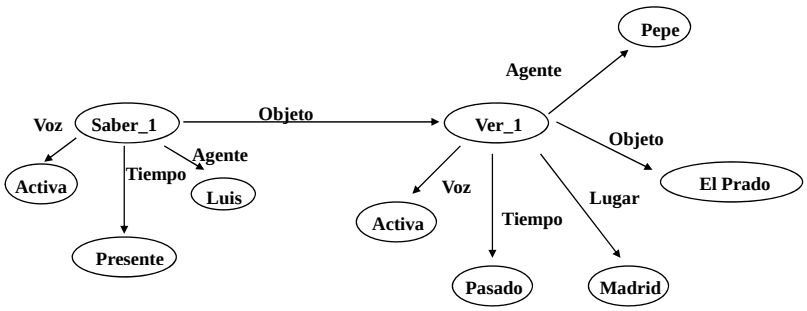
- **Objective**
  - Get used to the different syntaxes of RDF
- **Tasks**
  - Take the text of an RDF file and create its corresponding graph
  - Take an RDF graph and create its corresponding RDF/XML and N3 files

Exercise 1.a. Create a graph from a text file

- Open the file StickyNote_PureRDF.rdf
- Create the corresponding graph from it
- Compare your graph with those of your colleagues

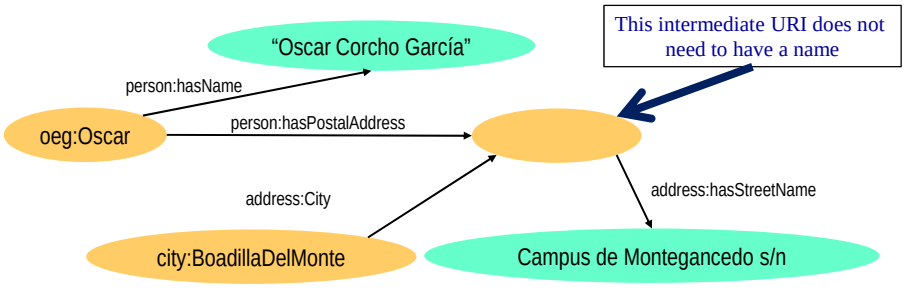
Exercise 1.b. Create RDF/XML and N3 text files from an RDF graph

- Transform the following graph into RDF/XML and N3 syntaxes



Blank nodes: structured property values

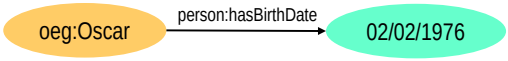
- Most real-world data involves structures that are more complicated than sets of RDF triple statements



- In RDF/XML, it is an <rdf:Description> node with no rdf:about
- In N3, it is an _:oscarAddress

Typed literals

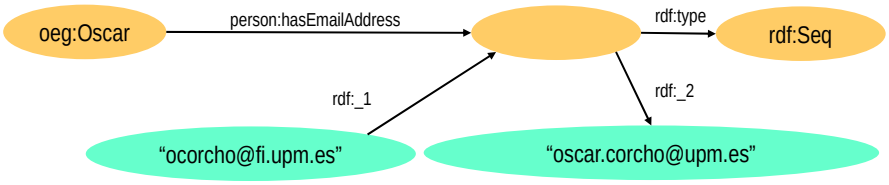
- **So far, all values have been presented as strings**
- **XML Schema datatypes can be used to specify values (objects in some RDF triple statements)**



- **In RDF/XML, this is expressed as:**
  - <rdf:Description rdf:about="#Oscar">
    <person:hasBirthDate
    rdf:datatype="http://www.w3.org/2001/XMLSchema#date">02/02/1976
    </person:hasBirthDate>
    </rdf:Description>
- **In N3, this is expressed as:**
  - oeg:Oscar person:hasBirthDate "02/02/1976"^xsd:date .

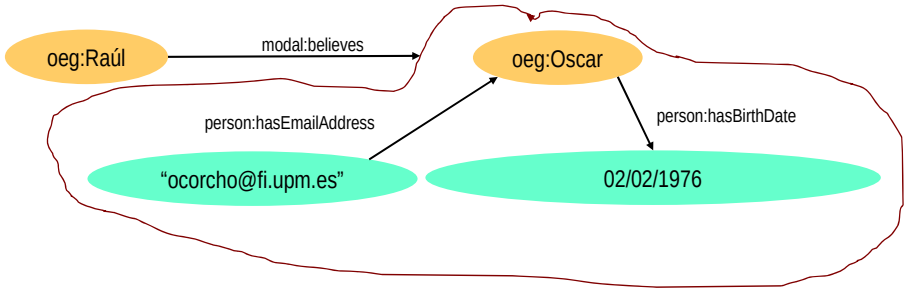
RDF Containers

- **There is often the need to describe groups of things**
  - A book was created by several authors
  - A lesson is taught by several persons
  - etc.
- **RDF provides a container vocabulary**
  - rdf:Bag → A group of resources or literals, possibly including duplicate members, where the order of members is not significant.
  - rdf:Seq → A group of resources or literals, possibly including duplicate members, where the order of members is significant.
  - rdf:Alt → A group of resources or literals that are alternatives (typically for a single value of a property).



RDF Reification

- **RDF statements about other RDF statements**
  - “Raúl believes that Oscar’s birthdate is on Feb 2nd, 1976 and that his e-mail address is ocorcho@fi.upm.es”



- **RDF Reification**
  - Allows expressing beliefs (and other modalities)
  - Allows expressing trust models, digital signatures,etc.
  - Allows expressing metadata about metadata

Table of Contents

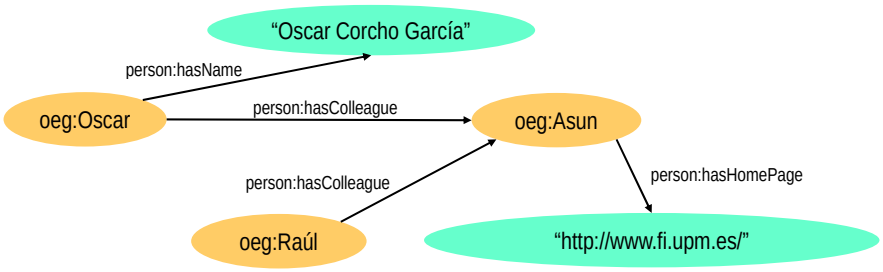
1. An introduction to knowledge representation formalisms
2. Resource Description Framework (RDF)
   - 2.1. RDF primitives
   - 2.2. Reasoning with RDF
3. RDF Schema
   - 3.1 RDF Schema primitives
   - 3.2 Reasoning with RDFS
4. RDF(S) management APIs
5. RDF(S) query languages: SPARQL

RDF inference. Graph matching techniques

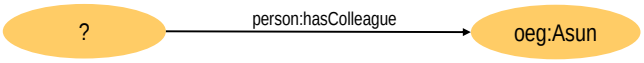
- **RDF inference is based on graph matching techniques**
- **Basically, the RDF inference process consists of the following steps:**
  - Transform an RDF query into a template graph that has to be matched against the RDF graph
    - It contains constant and variable nodes, and constant and variable edges between nodes.
  - Match against the RDF graph, taking into account constant nodes and edges.
  - Provide a solution for variable nodes and edges.

RDF inference. Examples (I)

- **Sample RDF graph**



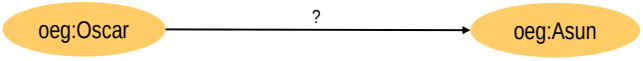
- **Query: "Tell me who are the persons who have Asun as a colleague"**



  - Result: oeg:Oscar and oeg:Raúl

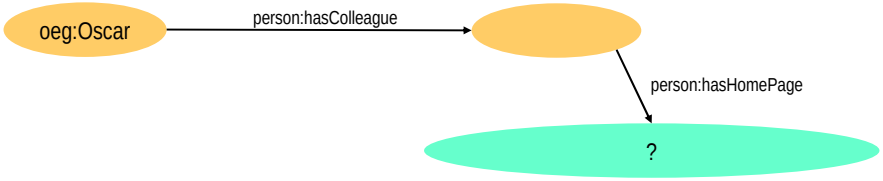
RDF inference. Examples (II)

- **Query: "Tell me which are the relationships between Oscar and Asun"**



  - Result: oeg:hasColleague

- **Query: "Tell me the homepage of Oscar colleagues"**



  - Result: "http://www.fi.upm.es/"

RDF inference. Entailment rules

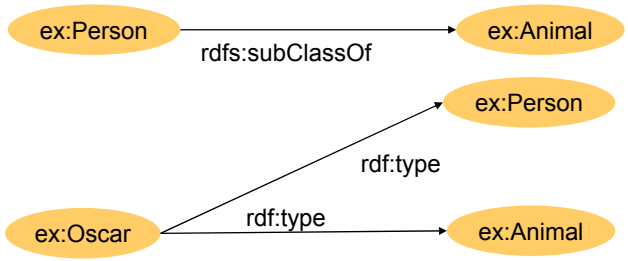
Rule Name	if E contains	then add
rdf1	uuu aaa yyy .	aaa `rdf:type rdf:Property .`
rdf2	uuu aaa lll . where lll is a well-typed XML literal .	_:nnn `rdf:type rdf:XMLLiteral .` where _:nnn identifies a blank node allocated to lll by rule lg.

Table of Contents

1. An introduction to knowledge representation formalisms
2. Resource Description Framework (RDF)
   2.1. RDF primitives
   2.2. Reasoning with RDF
3. RDF Schema
   3.1 RDF Schema primitives
   3.2 Reasoning with RDFS
4. RDF(S) management APIs
5. RDF(S) query languages: SPARQL

RDFS: RDF Schema

- **W3C Recommendation**
- **RDF Schema extends RDF to enable talking about classes of resources, and the properties to be used with them.**
  - Class definition: rdfs:Class, rdfs:subClassOf
  - Property definition: rdfs:subPropertyOf, rdfs:range, rdfs:domain
  - Other primitives: rdfs:comment, rdfs:label, rdfs:seeAlso, rdfs:isDefinedBy
- **RDFS vocabulary adds constraints on models, e.g.:**
  - ∀x,y,z type(x,y) and subClassOf(y,z) → type(x,z)



RDF(S) = RDF + RDF Schema. RDF/XML syntax

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:person="http://www.ontologies.org/ontologies/people#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns="http://www.oeg-upm.net/ontologies/people#"
   xml:base="http://www.oeg-upm.net/ontologies/people">
  <rdfs:Class rdf:about="http://www.ontologies.org/ontologies/people#Professor">
    <rdfs:subClassOf>
      <rdfs:Class rdf:about="http://www.ontologies.org/ontologies/people#Person"/>
    </rdfs:subClassOf>
   </rdfs:Class>
   <rdfs:Class rdf:about="http://www.ontologies.org/ontologies/people#Lecturer">
    <rdfs:subClassOf rdf:resource="http://www.ontologies.org/ontologies/people#Person"/>
   </rdfs:Class>
   <rdfs:Class rdf:about="http://www.ontologies.org/ontologies/people#PhDStudent">
     <rdfs:subClassOf rdf:resource="http://www.ontologies.org/ontologies/people#Person"/>
   </rdfs:Class>

  …
```

RDF(S) = RDF + RDF Schema. RDF/XML syntax

```
…
 <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasHomePage"/>
 <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasColleague">
  <rdfs:domain rdf:resource=" http://www.ontologies.org/ontologies/people#Person"/>
  <rdfs:range rdf:resource=" http://www.ontologies.org/ontologies/people#Person"/>
 </rdf:Property>
 <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasName">
  <rdfs:domain rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
 </rdf:Property>

 <person:PhDStudent rdf:ID="Raúl"/>
 <person:Professor rdf:ID="Asun">
    <person:hasColleague rdf:resource="#Raúl"/>
    <person:hasHomePage>http://www.fi.upm.es</person:hasHomePage>
 </person:Professor>
 <person:Lecturer rdf:ID="Oscar">
    <person:hasColleague rdf:resource="#Asun"/>
    <person:hasName>Oscar Corcho García</person:hasName>
 </person:Lecturer>
</rdf:RDF>
```


RDF(S) Serialisations. N3

```
@base  <http://www.oeg-upm.net/ontologies/people >
@prefix person: <http://www.ontologies.org/ontologies/people#>
person:hasColleague        a rdf:Property;
                           rdfs:domain person:Person;
                           rdfs:range person:Person.
person:Professor rdfs:subClassOf person:Person.
person:Lecturer rdfs:subClassOf person:Person.
person:PhDStudent rdfs:subClassOf person:Person.
:Asun     a person:Professor;
          person:hasColleague :Raúl ;
          person:hasHomePage "http://www.fi.upm.es/".
:Oscar    a person:Lecturer;
          person:hasColleague :Asun ;
          person:hasName "Oscar Corcho García".
:Raúl     a person:PhDStudent.
```

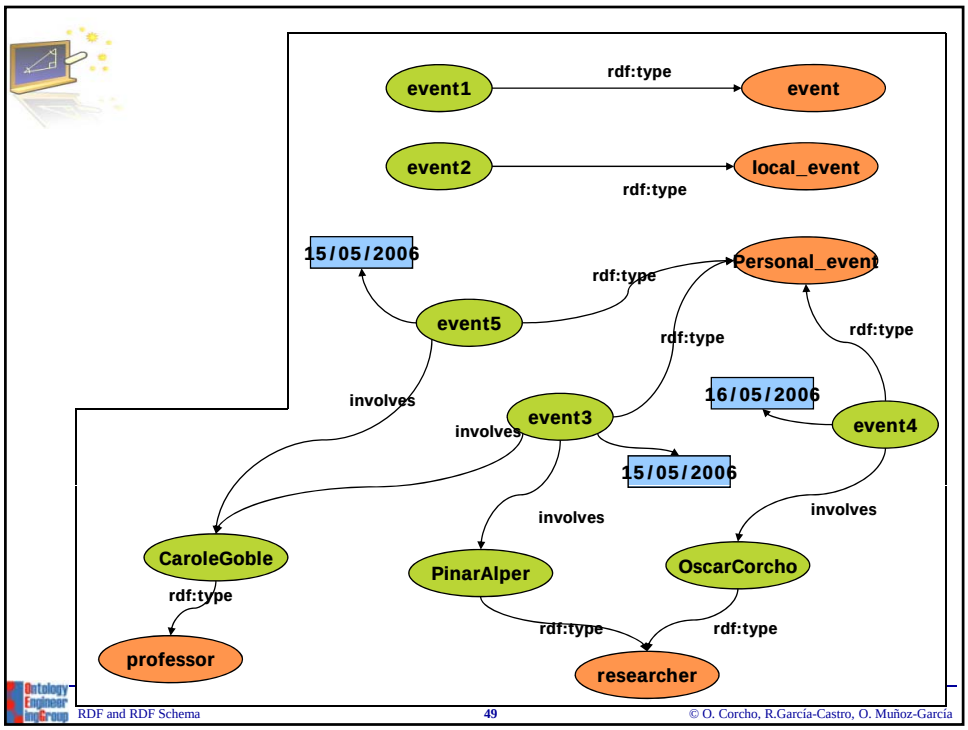
a is equivalent to rdf:type

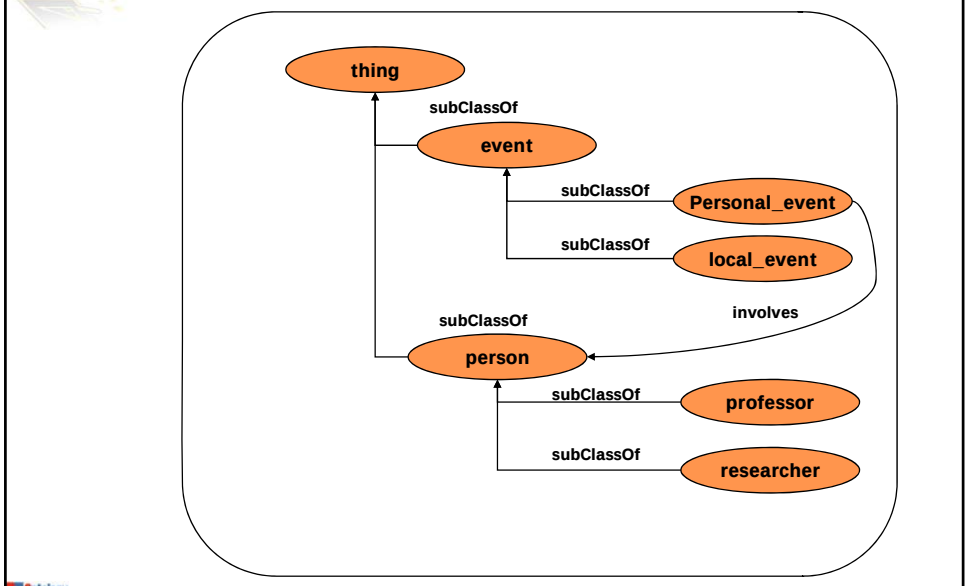
Exercise

- **Objective**
  - **Get used to the different syntaxes of RDF(S)**
- **Tasks**
  - **Take the text of an RDF(S) file and create its corresponding graph**
  - **Take an RDF(S) graph and create its corresponding RDF/XML and N3 files**

Exercise 2.a. Create a graph from a text file

- **Open the files StickyNote.rdf and StickyNote.rdfs**
- **Create the corresponding graph from them**
- **Compare your graph with those of your colleagues**


event1
rdf:type
event
event2
local_event
rdf:type
15/05/2006
Personal_event
rdf:type
event5
rdf:type
rdf:type
involves
16/05/2006
event3
event4
involves
15/05/2006
involves
involves
CaroleGoble
PinarAlper
OscarCorcho
rdf:type
rdf:type
rdf:type
professor
researcher


thing
subClassOf
event
subClassOf
Personal_event
subClassOf
local_event
involves
subClassOf
person
subClassOf
professor
subClassOf
researcher

Exercise 2.b. Create RDF/XML and N3 text files from an RDF(S) graph

- Transform the following graph into RDF/XML and N3 syntaxes

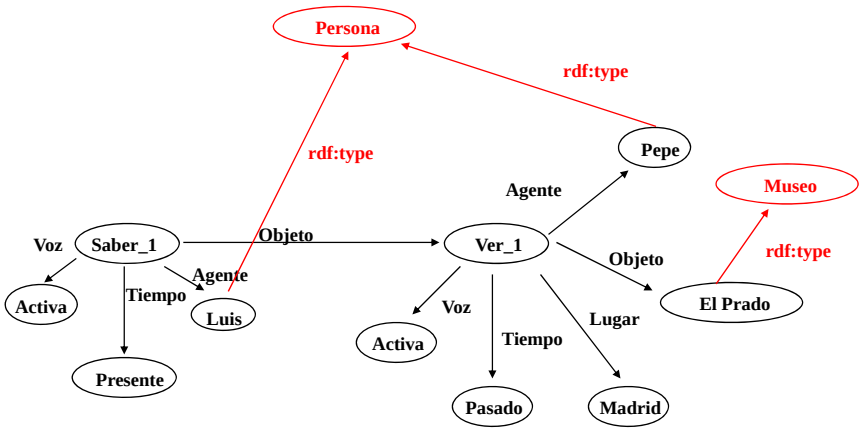


Table of Contents

1. An introduction to knowledge representation formalisms
2. Resource Description Framework (RDF)
   2.1. RDF primitives
   2.2. Reasoning with RDF
3. RDF Schema
   3.1 RDF Schema primitives
   3.2 Reasoning with RDFS
4. RDF(S) management APIs
5. RDF(S) query languages: SPARQL

RDF(S) inference. Entailment rules

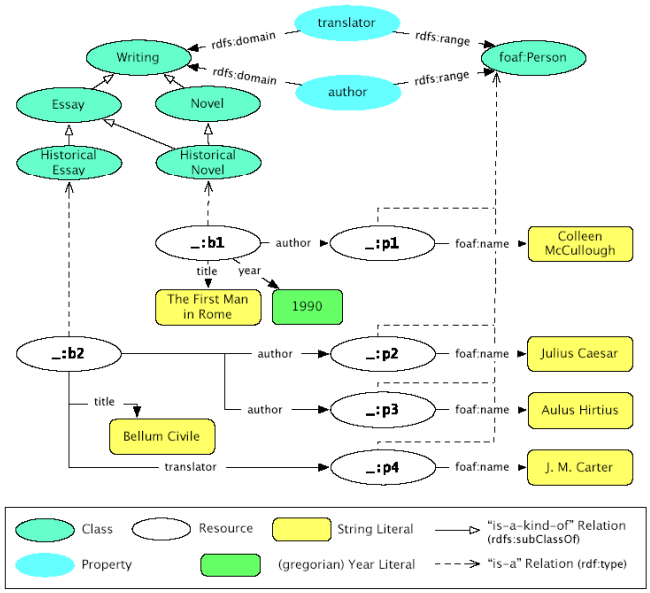
Rule Name	If E contains:	then add:
rdfs1	uuu aaa lll. where lll is a plain literal (with or without a language tag).	_:nnn rdf:type rdfs:Literal . where _:nnn identifies a blank node allocated to lll by rule rule lg.
rdfs2	aaa rdfs:domain xxx . uuu aaa yyy .	uuu rdf:type xxx .
rdfs3	aaa rdfs:range xxx . uuu aaa vvv .	vvv rdf:type xxx .
rdfs4a	uuu aaa xxx .	uuu rdf:type rdfs:Resource .
rdfs4b	uuu aaa vvv.	vvv rdf:type rdfs:Resource .
rdfs5	uuu rdfs:subPropertyOf vvv . vvv rdfs:subPropertyOf xxx .	uuu rdfs:subPropertyOf xxx .
rdfs6	uuu rdf:type rdf:Property .	uuu rdfs:subPropertyOf uuu .
rdfs7	aaa rdfs:subPropertyOf bbb . uuu aaa yyy .	uuu bbb yyy .
rdfs8	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf rdfs:Resource .
rdfs9	uuu rdfs:subClassOf xxx . vvv rdf:type uuu .	vvv rdf:type xxx .
rdfs10	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf uuu .
rdfs11	uuu rdfs:subClassOf vvv . vvv rdfs:subClassOf xxx .	uuu rdfs:subClassOf xxx .
rdfs12	uuu rdf:type rdfs:ContainerMembershipProperty .	uuu rdfs:subPropertyOf rdfs:member .
rdfs13	uuu rdf:type rdfs:Datatype .	uuu rdfs:subClassOf rdfs:Literal .

RDF(S) inference. Additional inferences

ext1	uuu rdfs:domain vvv . vvv rdfs:subClassOf zzz .	uuu rdfs:domain zzz .
ext2	uuu rdfs:range vvv . vvv rdfs:subClassOf zzz .	uuu rdfs:range zzz .
ext3	uuu rdfs:domain vvv . www rdfs:subPropertyOf uuu .	www rdfs:domain vvv .
ext4	uuu rdfs:range vvv . www rdfs:subPropertyOf uuu .	www rdfs:range vvv .
ext5	rdf:type rdfs:subPropertyOf www . www rdfs:domain vvv .	rdfs:Resource rdfs:subClassOf vvv .
ext6	rdfs:subClassOf rdfs:subPropertyOf www . www rdfs:domain vvv .	rdfs:Class rdfs:subClassOf vvv .
ext7	rdfs:subPropertyOf rdfs:subPropertyOf www . www rdfs:domain vvv .	rdf:Property rdfs:subClassOf vvv .
ext8	rdfs:subClassOf rdfs:subPropertyOf www . www rdfs:range vvv .	rdfs:Class rdfs:subClassOf vvv .
ext9	rdfs:subPropertyOf rdfs:subPropertyOf www . www rdfs:range vvv .	rdf:Property rdfs:subClassOf vvv .

Exercise

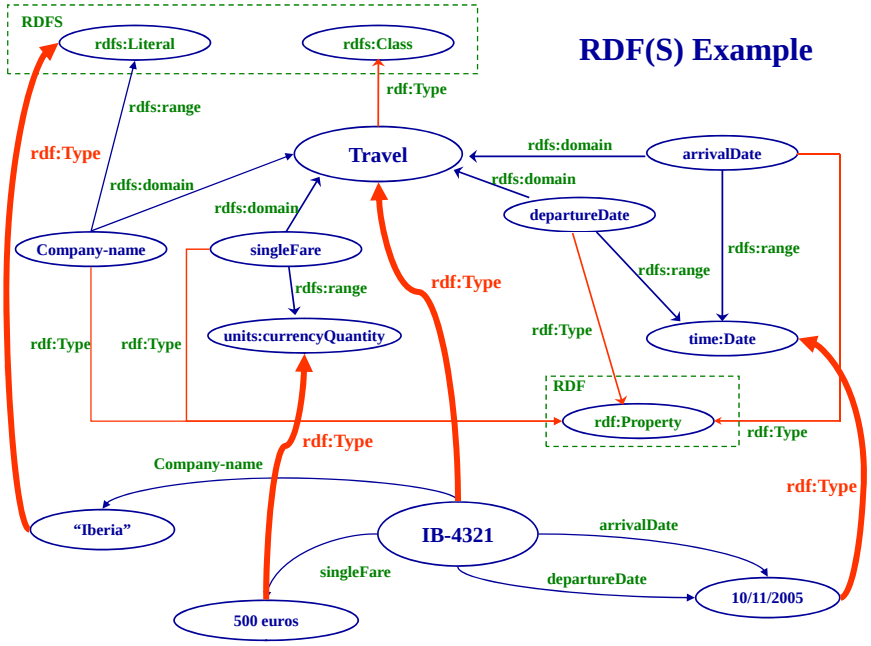
- **Objective**
  - **Understand how to RDF(S) reasoning works**
- **Tasks**
  - **Generate all possible RDF triples that can be inferred from an RDF(S) graph**



Exercise

- **Objective**
  - Understand how to use an RDF(S) development tool to create RDF(S) ontologies
- **Tasks**
  - Create the previous ontologies in an RDF(S) development tool

RDF(S) Example



Exercise

- **Objective**
  - **Understand the features of RDF(S) for implementing ontologies, including its limitations**
- **Tasks**
  - **Take the ontologies previously defined and create their graphs**
    - **First only include the vocabulary from the domain**
    - **Then include references to the RDF and RDFS vocabularies**

Domain description

- **Un lugar puede ser un lugar de interés.**
- **Los lugares de interés pueden ser lugares turísticos o establecimientos, pero no las dos cosas a la vez.**
- **Los lugares turísticos pueden ser palacios, iglesias, ermitas y catedrales.**
- **Los establecimientos pueden ser hoteles, hostales o albergues.**
- **Un lugar está situado en una localidad, la cual a su vez puede ser una villa, un pueblo o una ciudad.**
- **Un lugar de interés tiene una dirección postal que incluye su calle y su número.**
- **Las localidades tienen un número de habitantes.**
- **Las localidades se encuentran situadas en provincias.**

- **Covarrubias es un pueblo con 634 habitantes de la provincia de Burgos.**
- **El restaurante "El Galo" está situado en Covarrubias, en la calle Mayor, número 5.**
- **Una de las iglesias de Covarrubias está en la calle de Santo Tomás.**

Sample resulting ontology