RDF and RDF Schema

Oscar Corcho, Raúl García-Castro, Oscar Muñoz-García
({ocorcho,rgarcia,omunoz}@fi.upm.es)
Universidad Politécnica de Madrid

Acknowledgements: Axel Polleres, Mariano Fernández-López

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/

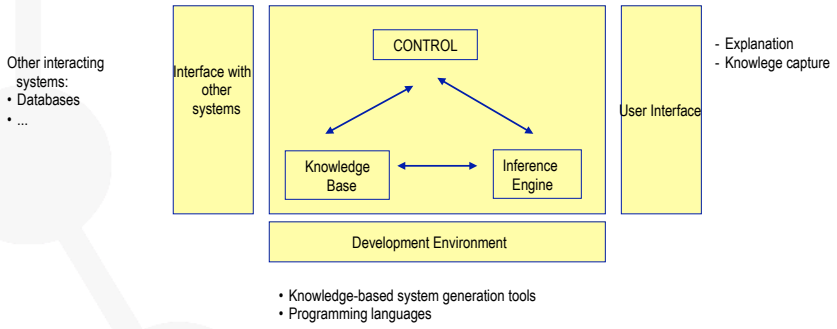
Jena web site: *http://jena.sourceforge.net/*
Jena API: *http://jena.sourceforge.net/tutorial/RDF_API/*
Jena tutorials: *http://www.ibm.com/developerworks/xml/library/j-jena/index.html*
http://www.xml.com/pub/a/2001/05/23/jena.html

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/

Index

- **An introduction to knowledge representation formalisms**
- Resource Description Framework (RDF)
  - **RDF primitives**
  - Reasoning with RDF
- RDF Schema
  - RDF Schema primitives
  - Reasoning with RDFS
- RDF(S) Management APIs

Architecture of a Knowledge-based System



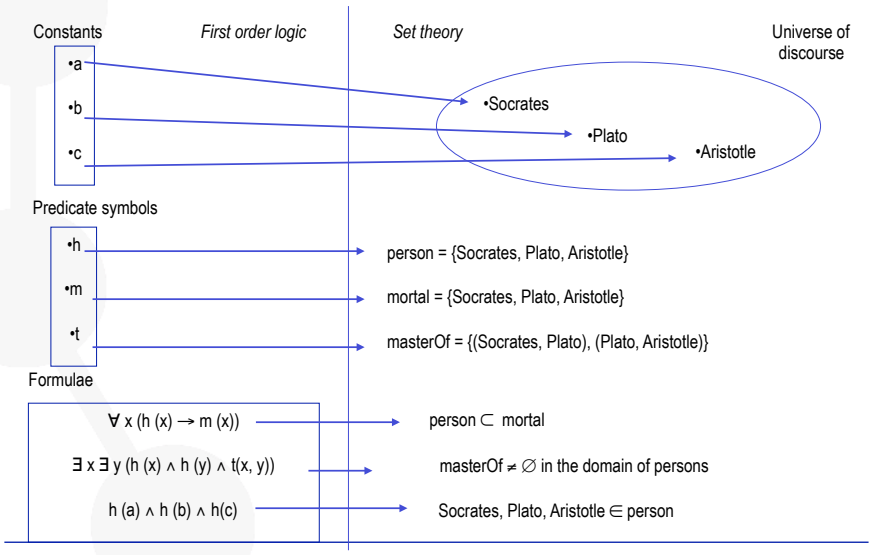
Knowledge Representation Formalisms. A Summary

- 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.

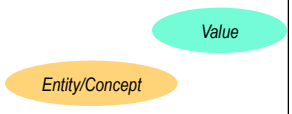
      $box(b) \wedge inside(b,h27)$
    - iii) Box B is not smaller than box A.

      $\neg smallerThan(b,a)$
  - We want to test whether box A is in room 27.

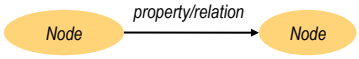
    $inside(a,h27)?$

Semantic Network. Basic elements

- Nodes
  - They represent entities or concepts, or values



- Edges
  - They represent properties or relations



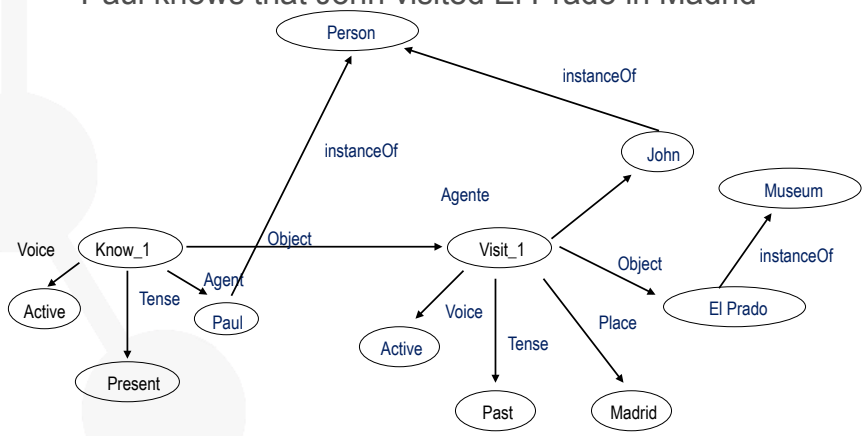
- The semantics (mapping to the real world) depends on the tags used for nodes and edges
- There is no predefined KR vocabulary
  - Although sometimes there are *structural* edges

Semantic networks. Example

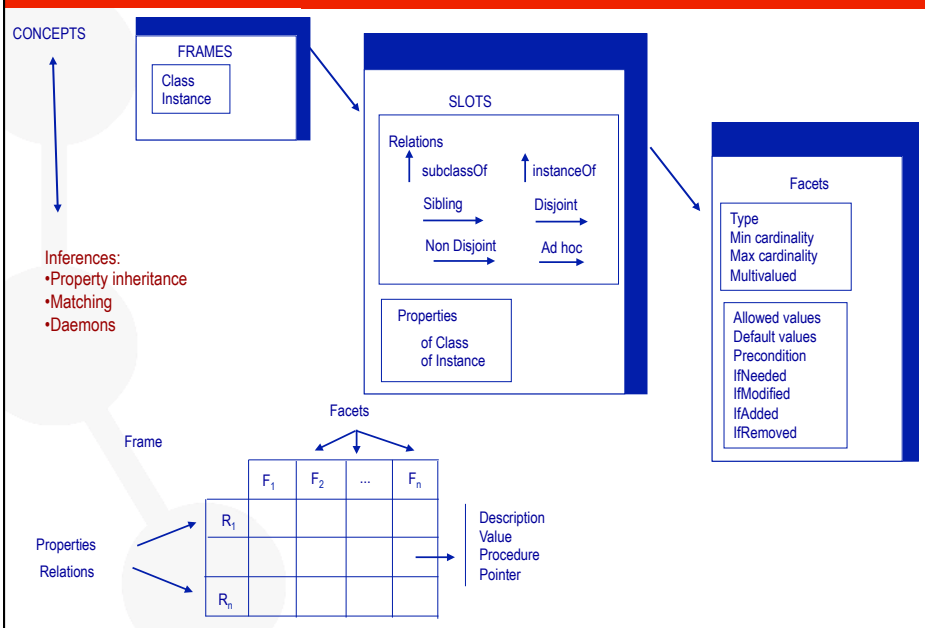
- Paul and John are persons
- El Prado is a museum
- Paul knows that John visited El Prado in Madrid

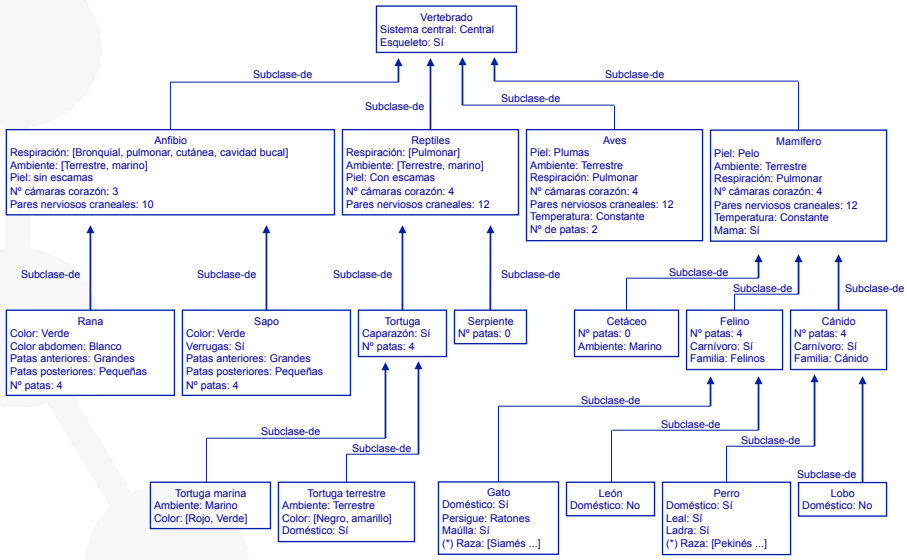
Semantic networks. Example. Solution

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



Frames. Basic elements



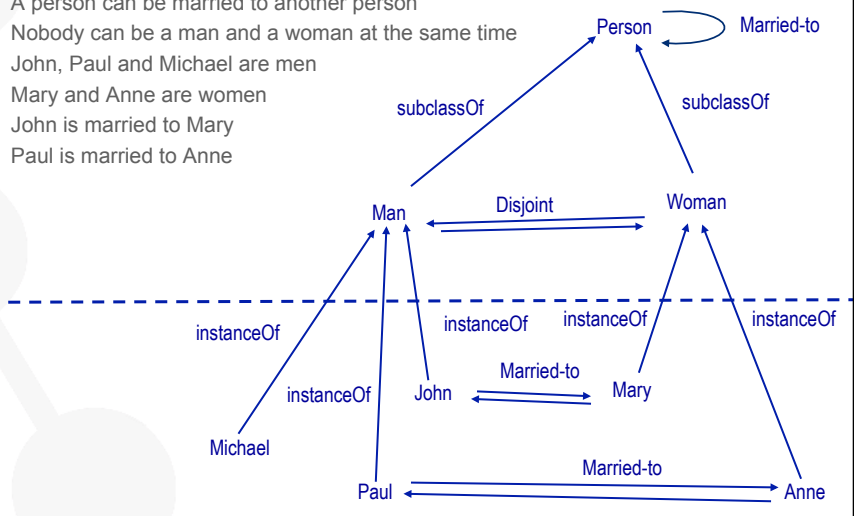


Frames. Example

- 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

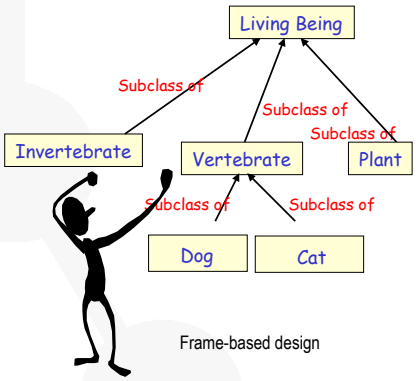
Frames. Example

- 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

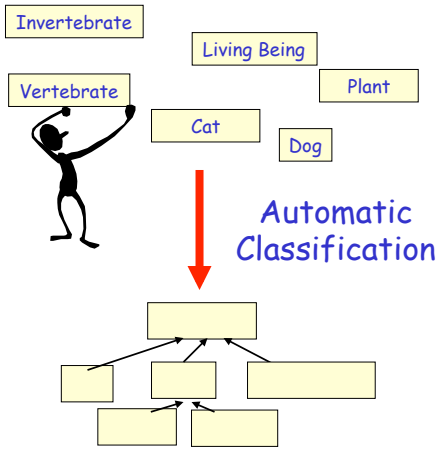


Description Logics. Basic elements

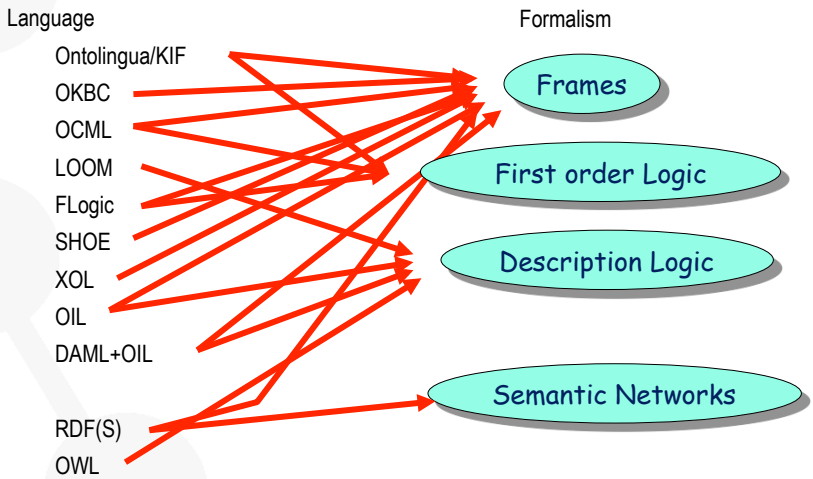
- A subset of first order logic with good reasoning properties
- Automatic classification



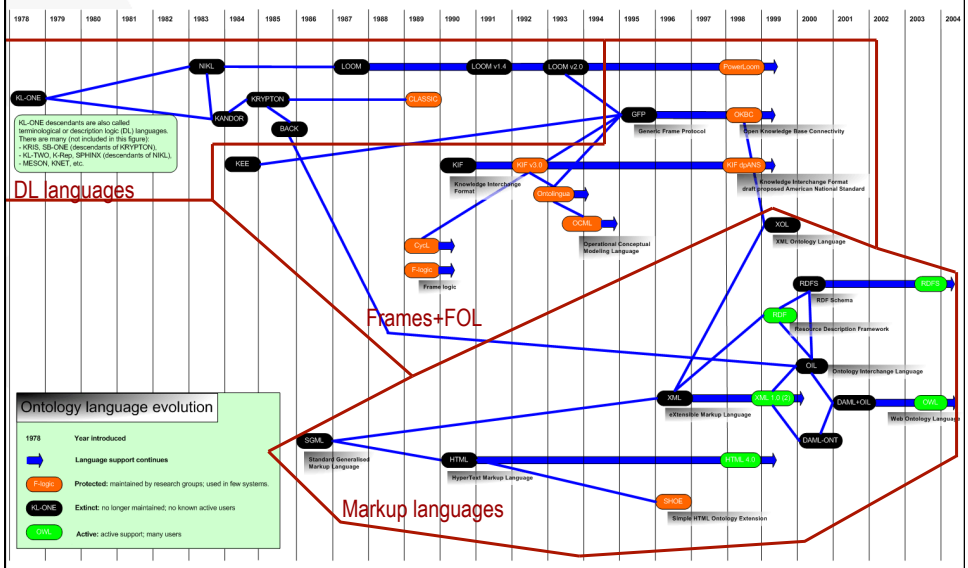
Frame-based design



KR Formalisms



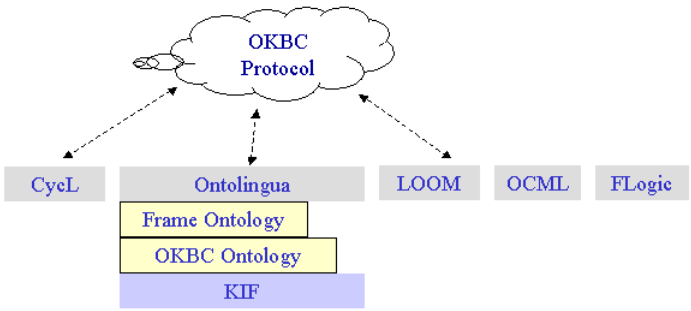
Ontology language evolution



Ontology Languages (I)

Traditional ontology languages

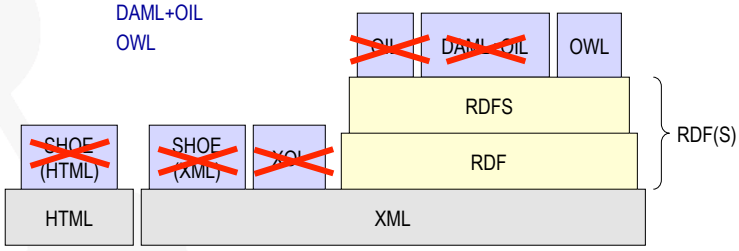
- Ontolingua/KIF
- OKBC
- OCML
- LOOM
- FLogic



Ontology Languages (II)

Ontology markup languages

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



Index

- An introduction to knowledge representation formalisms
- Resource Description Framework (RDF)
  - **RDF primitives**
  - Reasoning with RDF
- RDF Schema
  - RDF Schema primitives
  - Reasoning with RDFS
- RDF(S) Management APIs

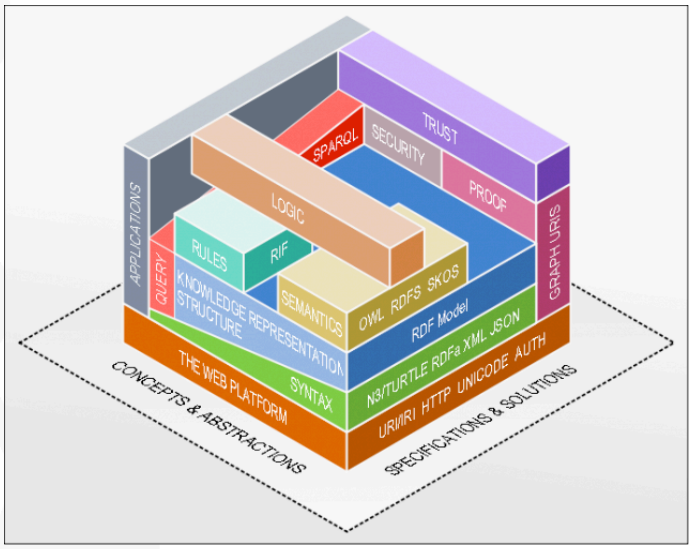
RDF and RDF Schema

- RDF: Resource Description Framework
- Goal
  - To describe the semantics of information in a machine-processable way

	Database	XML	RDF(S)	
Schema				RDF Schema
Data				RDF

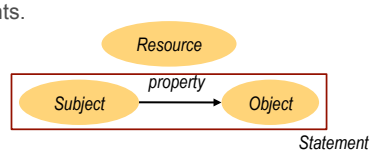
- W3C recommendation
  - Model
  - Syntax
  - Semantics

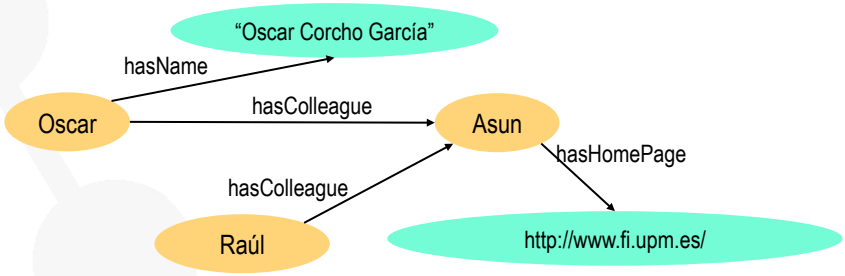
RDF(S) in the Semantic Web



RDF: Resource Description Framework

- 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]

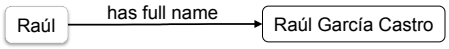




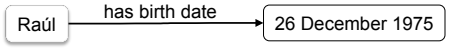
RDF statements

- Also known as "triples"
  - [Subject, Predicate, Object]
- "Raúl is a member of the Ontology Engineering Group"
  - [Raúl, is member of, Ontology Engineering Group]

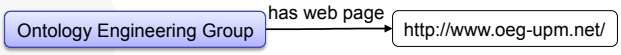
- "Raúl's full name is Raúl García Castro"
  - [Raúl, has full name, Raúl García Castro]



- "Raúl was born on 26 December 1975"
  - [Raúl, was born, 26 December 1975]

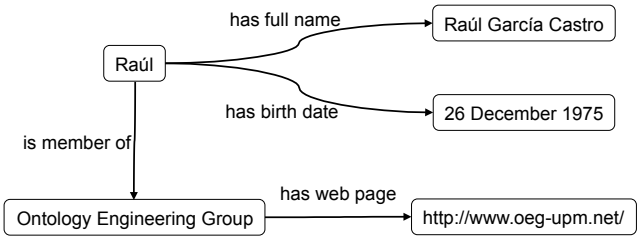


- "The Ontology Engineering Group web page is http://www.oeg-upm.net/"
  - [Ontology Engineering Group, has web page, http://www.oeg-upm.net/]



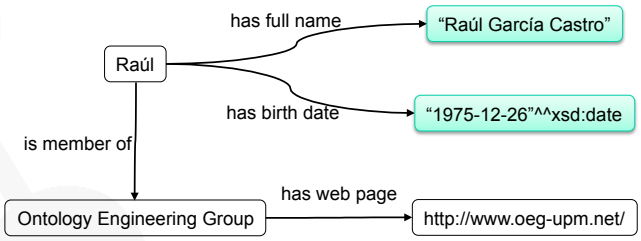
RDF graphs

- RDF graphs are sets of triples



RDF literals

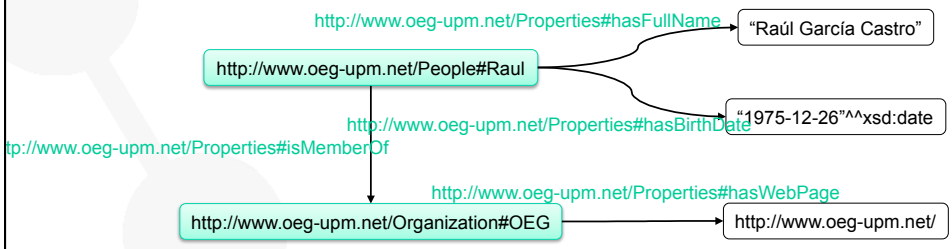
- Triple objects can be literals (character strings)
  - Subject and predicates are always resources
- Literals can be typed
  - Usually using XML Schema datatypes
  - RDF provides the ***rdf:XMLLiteral*** datatype



URIs in RDF

- URI component parts (RFC3986)
  - http://www.oeg-upm.net:8080/Info/People?position=current#Raul

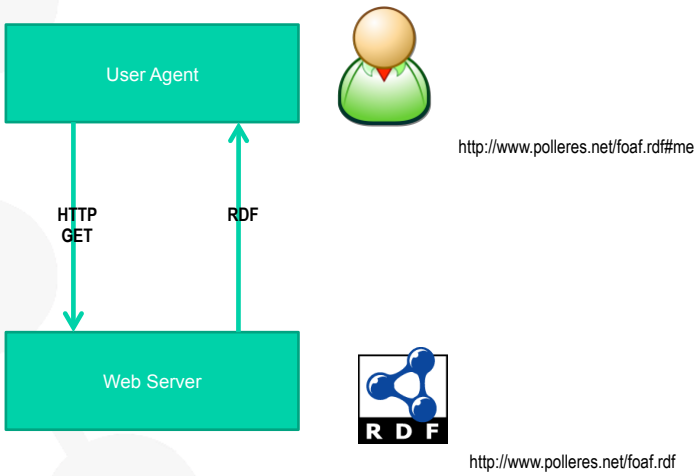
    Scheme | Authority | Path | Query | Fragment
- RDF URIs:
  - Are URI references: URI + Fragment
  - Can contain Unicode characters
  - Identify resources and values (e.g., mailto:rgarcia@fi.upm.es)



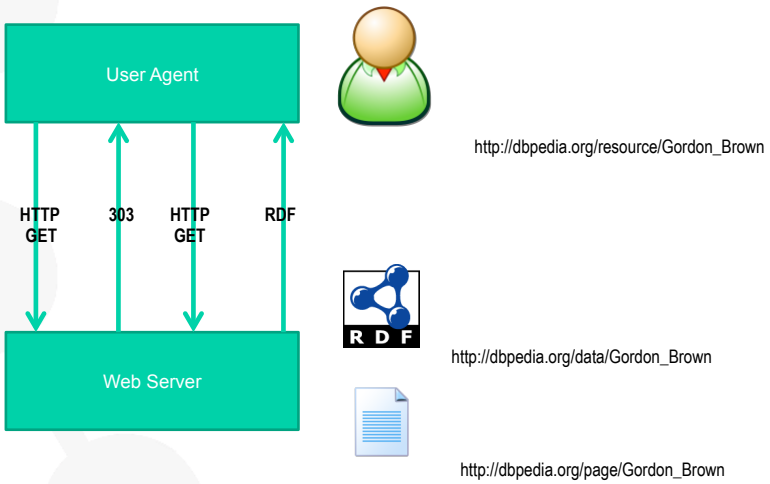
Types of URIs

- Thing-URIs, Hash URIs or URIRefs (Unique Resource Identifiers References)
  - A URI and an optional Fragment Identifier separated from the URI by the hash symbol '#'
  - http://www.ontology.org/people#Person
  - people:Person
- Source URIs or Slash URIs can also be used, as in FOAF:
  - http://xmlns.com/foaf/0.1/Person

Correspondence between thing-URI and source-URI



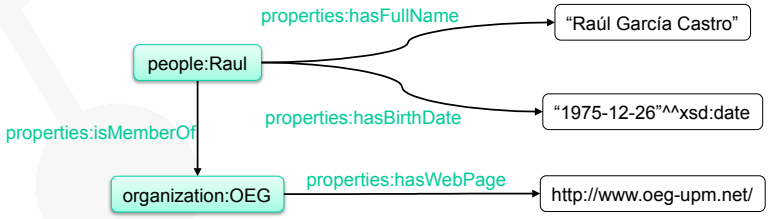
Correspondence between thing-URI and source-URI



Namespaces in RDF

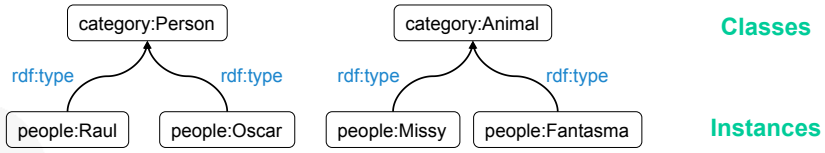
- Namespaces defined using XML qualified names
- URIs under a namespace are called vocabularies

Prefix	URI
people	http://www.oeg-upm.net/People#
organization	http://www.oeg-upm.net/Organization#
properties	http://www.oeg-upm.net/Properties#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#

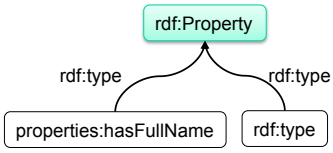


Classifying resources

- The ***rdf:type*** property is used to classify resources in categories/classes

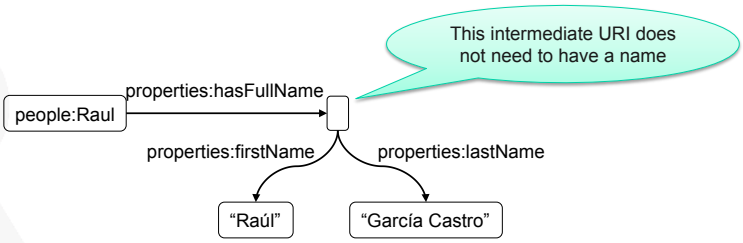


- The ***rdf:Property*** class is the class of all properties



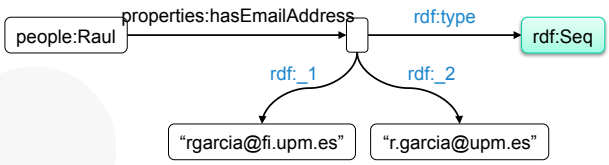
Blank nodes: structured property values

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



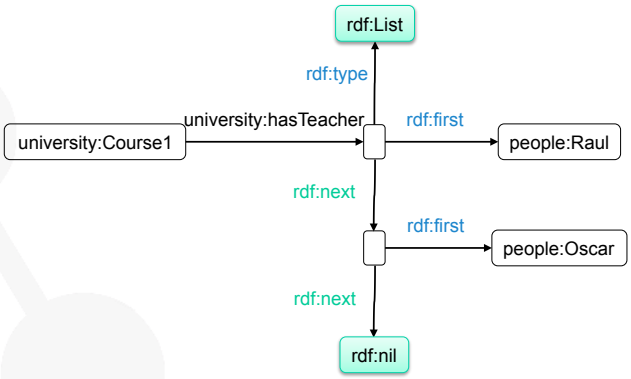
RDF Containers

- 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.*** Group of resources or literals, including duplicates, where order is not significant
  - ***rdf:Seq.*** Group of resources or literals, including duplicates, where order is significant
  - ***rdf:Alt.*** Group of resources or literals that are alternatives (typically for a single value of a property)



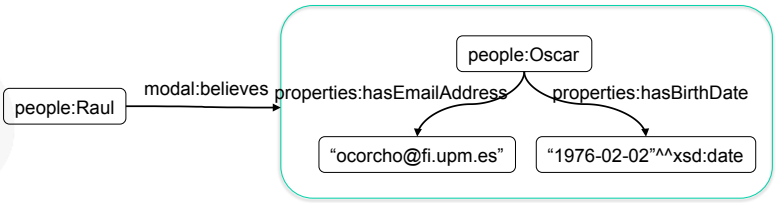
RDF Collections

- Groups of things represented as a list structure
  - "A container with limits"
- Constructed using ***rdf:List***, ***rdf:first***, ***rdf:rest***, and ***rdf:nil***



RDF Reification

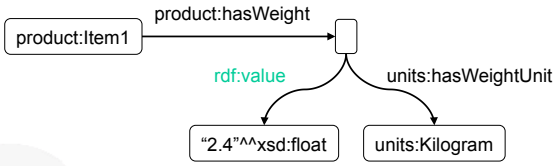
- RDF statements about other RDF statements
  - "Raúl believes that Oscar's birthdate is on Feb 2nd, 1976 and that his e-mail address is ocorcho@fi.upm.es"
- Expressed using ***rdf:Statement***, ***rdf:subject***, ***rdf:predicate***, and ***rdf:object***



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

Main value of a structured value

- Sometimes one of the values of a structured value is the main one
  - The weight of an item is 2.4 kilograms
  - The main value is 2.4, which is expressed with ***rdf:value***
- Scarcely used



RDF vocabulary summary

Classes	Properties	Individuals
Classification		
rdf:Property	*rdf:type*	
Containers		
rdf:Bag	*rdf:_1, rdf:_2, rdf:_3…*	
rdf:Seq		
rdf:Alt		
Collections		
rdf:List	*rdf:first*	*rdf:nil*
	rdf:rest	
Reification		
rdf:Statement	*rdf:subject*	
	rdf:predicate	
	rdf:object	
Values		
rdf:XMLLiteral	*rdf:value*	

RDF Serialisations

- Normative
  - **RDF/XML** (www.w3.org/TR/rdf-syntax-grammar/)
- Working Draft (9 August 2011)
  - **Turtle** (http://www.w3.org/TR/turtle/)
- Alternative (for human consumption)
  - **N3** (http://www.w3.org/DesignIssues/Notation3.html)
  - **TriX** (http://www.w3.org/2004/03/trix/)
  - …
- **Important:** the RDF serializations allow different syntactic variants.
  - E.g., the order of RDF statements has no meaning

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

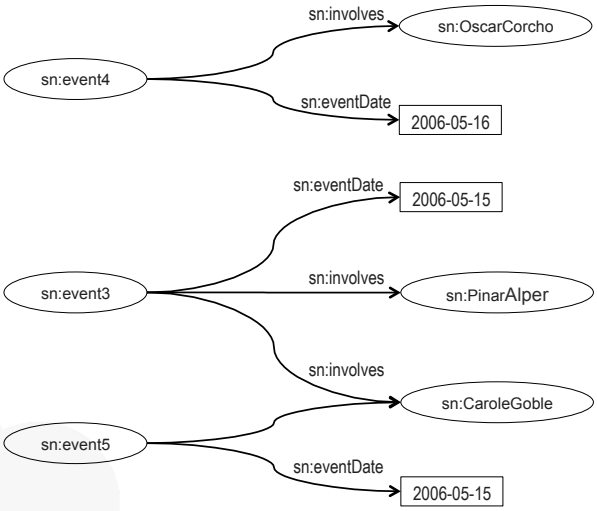
Exercise

- 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 file

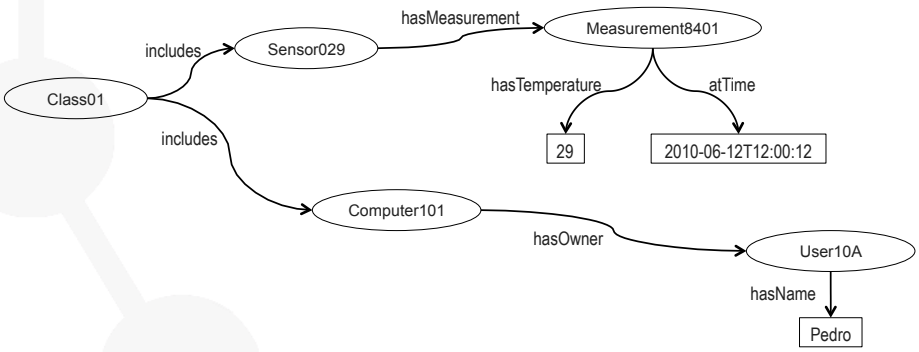
- Open the file StickyNote_PureRDF.rdf
- Create the corresponding graph from it
- Compare your graph with those of your colleagues

Exercise 1.a. StickyNote_PureRDF.rdf



Exercise 1.b. Create files from a graph

- Transform the following graph into N3 syntax



Index

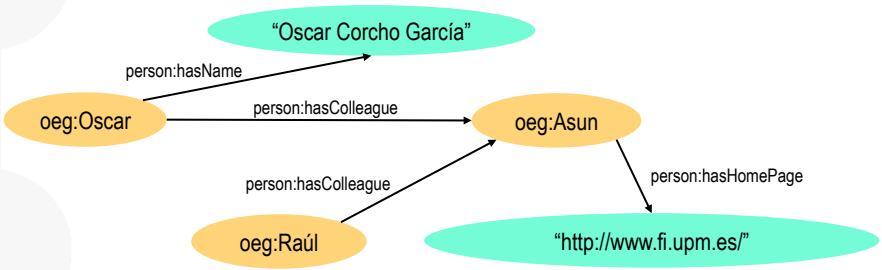
- An introduction to knowledge representation formalisms
- Resource Description Framework (RDF)
  - RDF primitives
  - **Reasoning with RDF**
- RDF Schema
  - RDF Schema primitives
  - Reasoning with RDFS
- RDF(S) Management APIs

RDF inference. Graph matching techniques

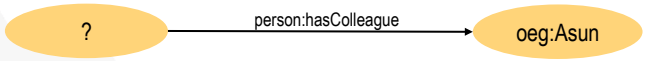
- RDF inference is based on graph matching techniques
- Basically, the RDF inference process consists of the following steps:
  - Transform an RDF query into a template graph that has to be matched against the RDF graph
    - It contains constant and variable nodes, and constant and variable edges between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



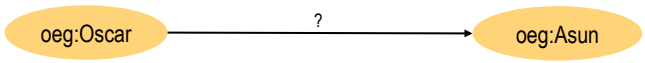
- **Query**: "Tell me who are the persons who have Asun as a colleague"



  - **Result**: oeg:Oscar and oeg:Raúl

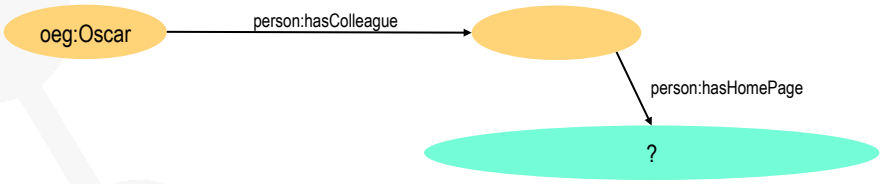
RDF inference. Examples (II)

- **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

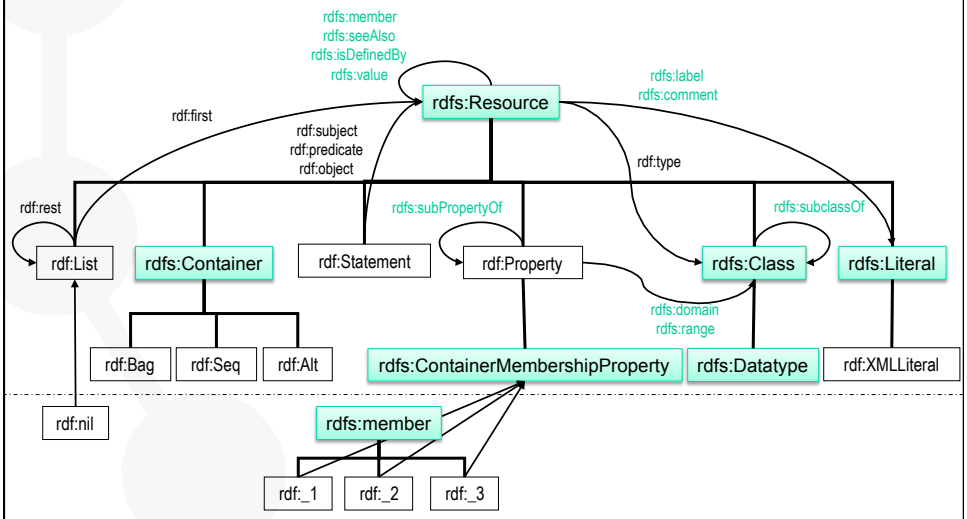
Rule Name	if E contains	then add
rdf1	uuu aaa yyy .	aaa `rdf:type rdf:Property .`
rdf2	uuu aaa lll . where lll is a well-typed XML literal .	_:nnn `rdf:type rdf:XMLLiteral .` where _:nnn identifies a blank node allocated to lll by rule lg.

Index

- An introduction to knowledge representation formalisms
- Resource Description Framework (RDF)
  - RDF primitives
  - Reasoning with RDF
- RDF Schema
  - **RDF Schema primitives**
  - Reasoning with RDFS
- RDF(S) Management APIs

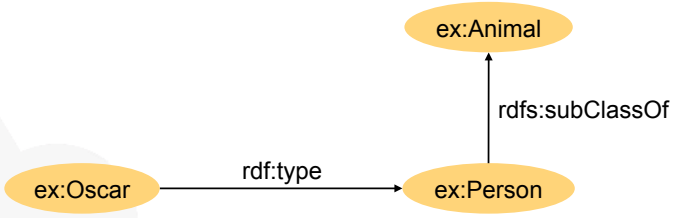
RDF Schema

- Extends RDF
- Allows describing classes of resources and their properties



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) Serialisations. 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#PhD">
     <rdfs:subClassOf rdf:resource="http://www.ontologies.org/ontologies/people#Person"/>
  </rdfs:Class>
 …
```


RDF(S) Serialisations. RDF/XML syntax

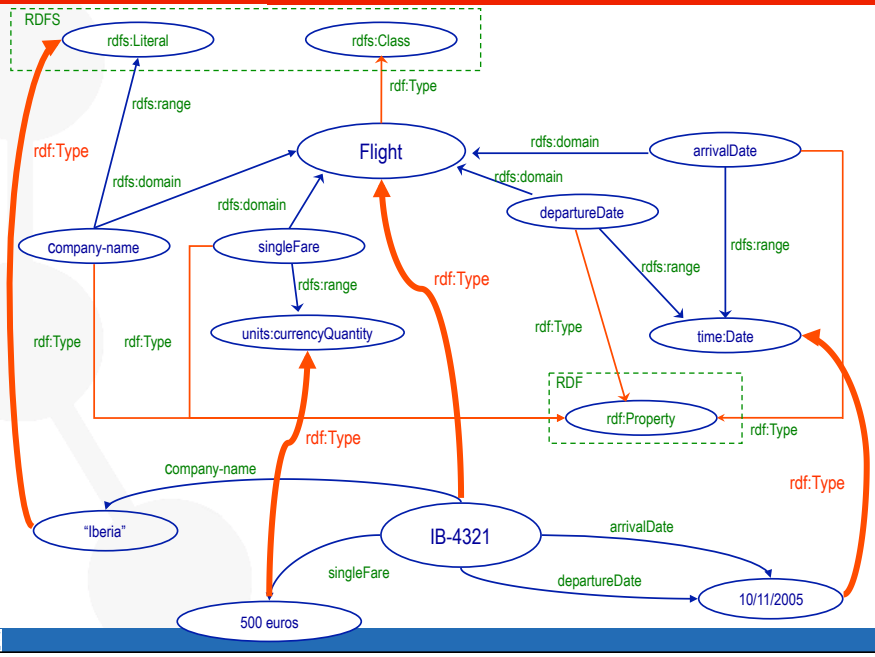
```
…
  <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasHomePage"/>
  <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasColleague">
    <rdfs:domain rdf:resource=" http://www.ontologies.org/ontologies/people#Person"/>
    <rdfs:range rdf:resource=" http://www.ontologies.org/ontologies/people#Person"/>
  </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:PhD rdf:ID="Raul"/>
  <person:Professor rdf:ID="Asun">
     <person:hasColleague rdf:resource="#Raul"/>
     <person:hasHomePage>http://www.fi.upm.es</person:hasHomePage>
  </person:Professor>
  <person:Lecturer rdf:ID="Oscar">
     <person:hasColleague rdf:resource="#Asun"/>
     <person:hasName>Óscar 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:PhD rdfs:subClassOf person:Person.
:Asun     a person:Professor;
          person:hasColleague :Raul ;
          person:hasHomePage "http://www.fi.upm.es/".
:Oscar    a person:Lecturer;
          person:hasColleague :Asun ;
          person:hasName "Óscar Corcho García".
:Raul     a person:PhD.
```

a is equivalent to rdf:type

RDF(S) Example

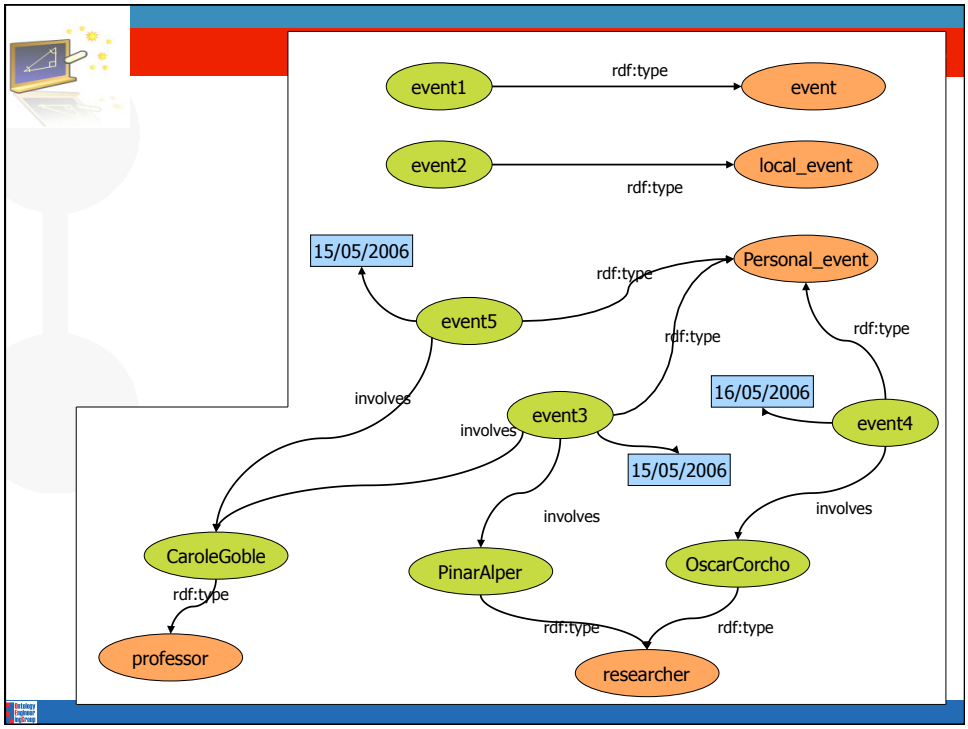


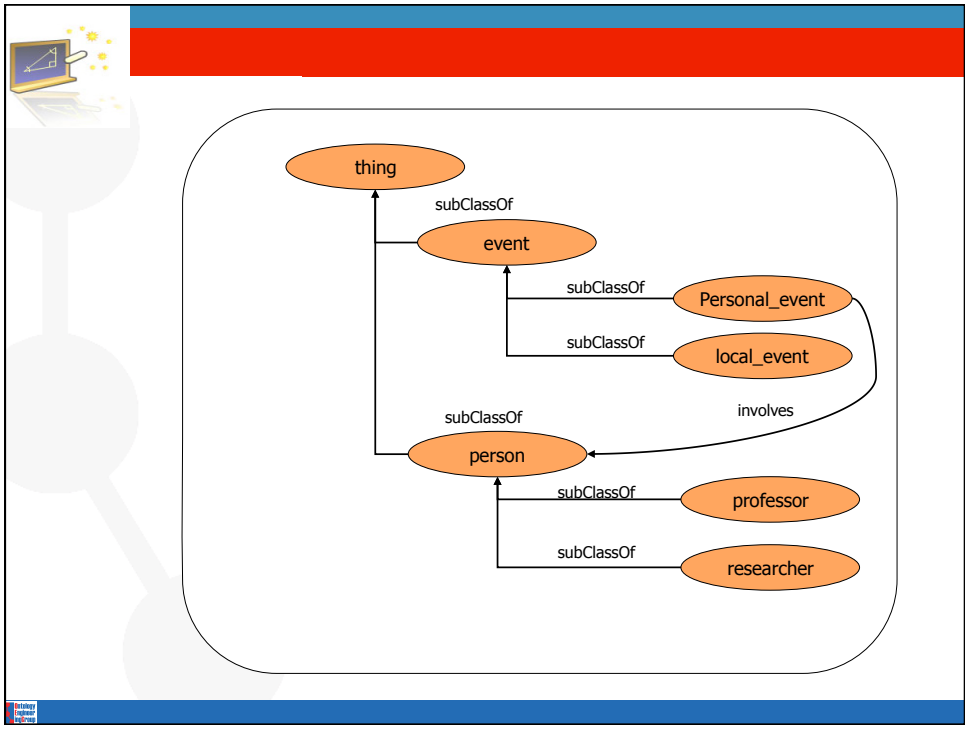
Exercise

- 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 file

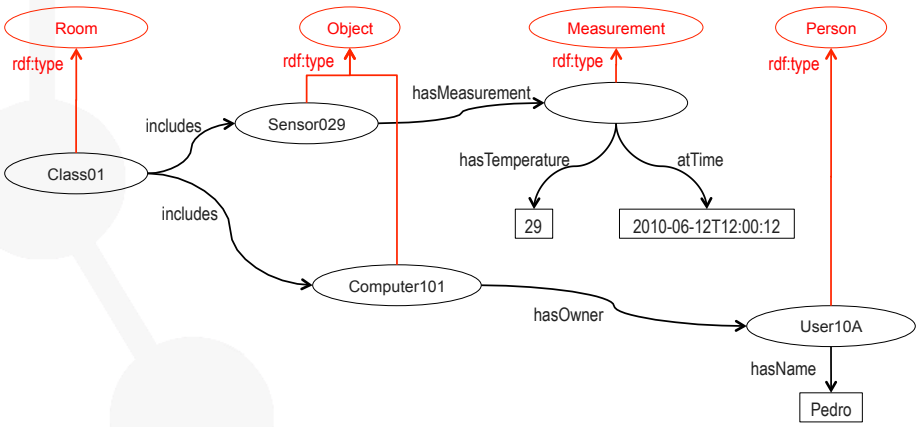
- Open the files StickyNote.rdf and StickyNote.rdfs
- Create the corresponding graph from them
- Compare your graph with those of your colleagues


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
subClassOf
involves
person
subClassOf
professor
subClassOf
researcher

Exercise 2.b. Create files from a graph

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



Index

- An introduction to knowledge representation formalisms
- Resource Description Framework (RDF)
  - RDF primitives
  - Reasoning with RDF
- RDF Schema
  - RDF Schema primitives
  - **Reasoning with RDFS**
- RDF(S) Management APIs

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 .

RDF(S) limitations

- RDFS too weak to describe resources in sufficient detail
  - No localised range and domain constraints
    - Can't say that the range of hasChild is person when applied to persons and elephant when applied to elephants
  - No existence/cardinality constraints
    - Can't say that all *instances* of person have a mother that is also a person, or that persons have exactly 2 parents
  - No boolean operators
    - Can't say or, not, etc.
  - No transitive, inverse or symmetrical properties
    - Can't say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
- Difficult to provide reasoning support
  - No "native" reasoners for non-standard semantics
  - May be possible to reason via FOL axiomatisation

Exercise

•Objective
- Understand the features of RDF(S) for implementing ontologies, including its limitations

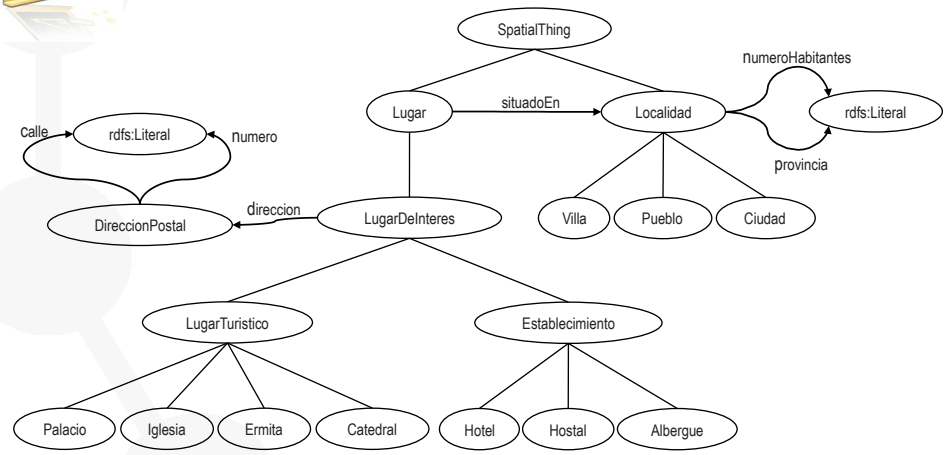
•Tasks
- From a domain description, create the RDF(S) graph
  - First only include the vocabulary from the domain
  - Then include references to the RDF and RDFS vocabularies

Exercise 3. 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.

Exercise 3. Sample resulting ontology



Index

- An introduction to knowledge representation formalisms
- Resource Description Framework (RDF)
  - RDF primitives
  - Reasoning with RDF
- RDF Schema
  - RDF Schema primitives
  - Reasoning with RDFS
- **RDF(S) Management APIs**

Sample RDF APIs

- RDF libraries for different languages:
  - Java, Python, C, C++, C#, .Net, Javascript, Tcl/Tk, PHP, Lisp, Obj-C, Prolog, Perl, Ruby, Haskell
  - List in
- Usually related to a RDF repository
- Multilanguage:
  - Redland RDF Application Framework (C, Perl, PHP, Python and Ruby): http://www.redland.opensource.ac.uk/
- Java:
  - Jena: http://jena.sourceforge.net/
  - Sesame: http://www.openrdf.org/
- PHP:
  - RAP - RDF API for PHP: http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/
- Python:
  - RDFLib: http://rdflib.net/
  - Pyrple: http://infomesh.net/pyrple/

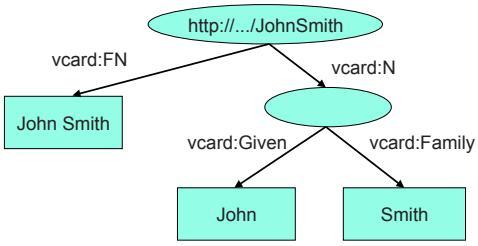
Jena

- Java framework for building Semantic Web applications
- Open source software from HP Labs
- The Jena framework includes:
  - A RDF API
  - An OWL API
  - Reading and writing RDF in RDF/XML, N3 and N-Triples
  - In-memory and persistent storage
  - A rule based inference engine
  - SPARQL query engine

Sesame

- A framework for storage, querying and inferencing of RDF and RDF Schema
- A Java Library for handling RDF
- A Database Server for (remote) access to repositories of RDF data
- Highly expressive query and transformation languages
  - SeRQL, SPARQL
- Various backends
  - Native Store
  - RDBMS (MySQL, Oracle 10, DB2, PostgreSQL)
  - main memory
- Reasoning support
  - RDF Schema reasoner
  - OWL DLP (OWLIM)
  - domain reasoning (custom rule engine)

Jena example. Graph creation




```
// some definitions
String personURI = "http://somewhere/JohnSmith";
String givenName = "John";
String familyName = "Smith";
String fullName = givenName + " " + familyName;
// create an empty
Model Model model = ModelFactory.createDefaultModel();
// create the resource
// and add the properties cascading style
Resource johnSmith = model.createResource(personURI)
    .addProperty(VCARD.FN, fullName)
    .addProperty(VCARD.N, model.createResource()
    .addProperty(VCARD.Given, givenName)
    .addProperty(VCARD.Family, familyName));
```


Jena example. Read and write

```
// create an empty model
Model model = ModelFactory.createDefaultModel();

// use the FileManager to find the input file
InputStream in = FileManager.get().open( inputFileName );
if (in == null) {
    throw new IllegalArgumentException("File not found");
}

// read the RDF/XML file
model.read(in, "");

// write it to standard out
model.write(System.out);
```

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'
 >
  <rdf:Description rdf:nodeID="A0">
    <vcard:Family>Smith</vcard:Family>
    <vcard:Given>John</vcard:Given>
  </rdf:Description>
  <rdf:Description rdf:about='http://somewhere/JohnSmith/'>
    <vcard:FN>John Smith</vcard:FN>
    <vcard:N rdf:nodeID="A0"/>
  </rdf:Description>
...
</rdf:RDF>
```

Some RDF editors

- IsaViz
  - http://www.w3.org/2001/11/IsaViz/
- Morla
  - http://www.morlardf.net/
- RDFAuthor
  - http://rdfweb.org/people/damian/RDFAuthor/
- RdfGravity
  - http://semweb.salzburgresearch.at/apps/rdf-gravity/
- Rhodonite
  - http://rhodonite.angelite.nl/

Main References

- 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/

- RDF validator:
  http://www.w3.org/RDF/Validator/
- RDF resources:
  http://planetrdf.com/guide/

Exercise

- Objective
  - Understand how to use an RDF(S) management API
- Tasks
  - Read an ontology in RDF(S) from two files:
    - GP_Santiago.rdf (conceptualization)
    - GP_Santiago.rdfs (instances)
  - Write the class hierarchy of the ontology, including the instances of each class

Hands-on

- Read an ontology in RDF(S) from two files:
  - GP_Santiago.rdf (conceptualization)
  - GP_Santiago.rdfs (instances)
- Write the class hierarchy of the ontology, including the instances of each class:

```
Class Practica2:MedioTransporte
  Class Practica2:Tren
  Class Practica2:Bicicleta
    Instance Practica2:GP_Santiago_Instance_70
  Class Practica2:Automovil
  Class Practica2:AutoBus
  Class Practica2:APie
Class Practica2:InfraEstructuraTransporte
  Class Practica2:ViaFerrea
  Class Practica2:Sendero
  Class Practica2:Carretera
    Instance Practica2:A6
...
```

RDF and RDF Schema

Oscar Corcho, Raúl García-Castro, Oscar Muñoz-García
({ocorcho,rgarcia,omunoz}@fi.upm.es)
Universidad Politécnica de Madrid

Acknowledgements: Axel Polleres, Mariano Fernández-López