Ontology Languages

Raúl García-Castro, Óscar Corcho

Ontology Engineering Group
Universidad Politécnica de Madrid, Spain

Speaker: Raúl García Castro
rgarcia@fi.upm.es

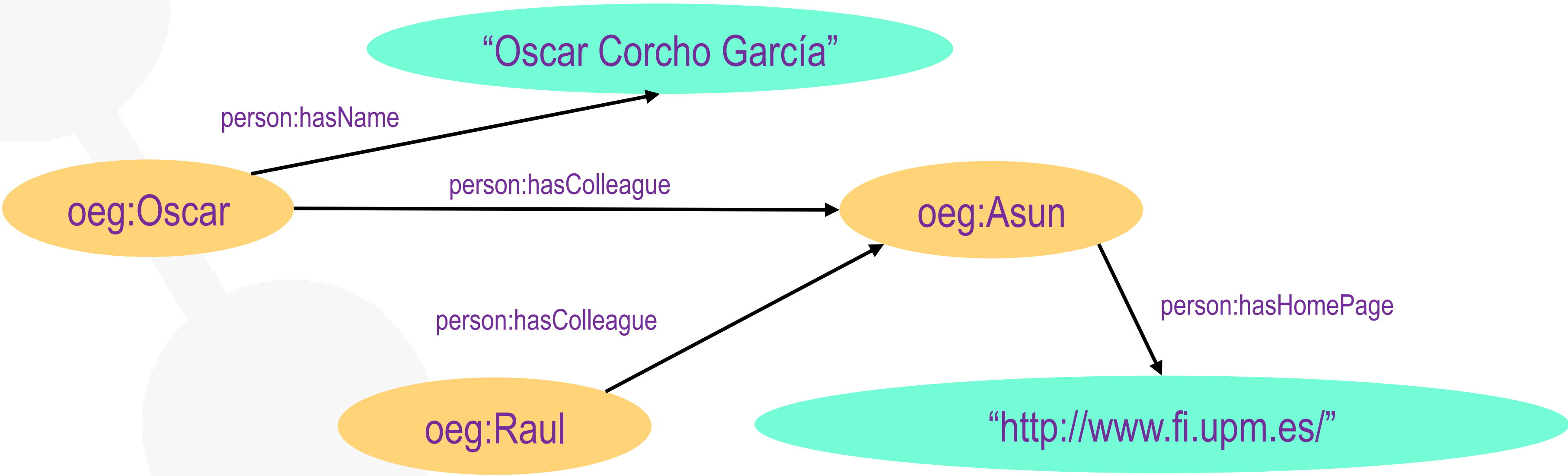
ATHENS 2010
Madrid, Spain

Index

- **Resource Description Framework (RDF)**
  - **RDF primitives**
  - Reasoning with RDF
- RDF Schema
  - RDF Schema primitives
  - Reasoning with RDFS
- RDF(S) Management APIs
- Web Ontology Language (OWL)
  - OWL primitives
  - Reasoning with OWL

RDF: Resource Description Framework

- W3C recommendation
- RDF is graphical formalism (+ XML syntax + semantics)
  - For representing metadata
  - For describing the semantics of information in a machine-accessible way
- RDF is a basic ontology language
  - 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 URIs

- RDF uses URIRefs (Uniform Resource Identifiers References) to identify resources
  - A URIRef consists of a URI and an optional Fragment Identifier separated from the URI by the hash symbol ‘#’
  - Examples
    - http://www.co-ode.org/people#hasColleague
    - coode:hasColleague
- A set of URIRefs is known as a vocabulary
  - E.g., the RDF Vocabulary
    - The set of URIRefs used in describing the RDF concepts: rdf:Property, rdf:Resource, rdf:type, etc.
  - The RDFS Vocabulary
    - The set of URIRefs used in describing the RDF Schema language: rdfs:Class, rdfs:domain, etc.
  - The ‘Pizza Ontology’ Vocabulary
    - pz:hasTopping, pz:Pizza, pz:VegetarianPizza, etc.

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: 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>
```

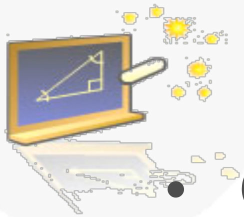
RDF Serialisations. N3

```
@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

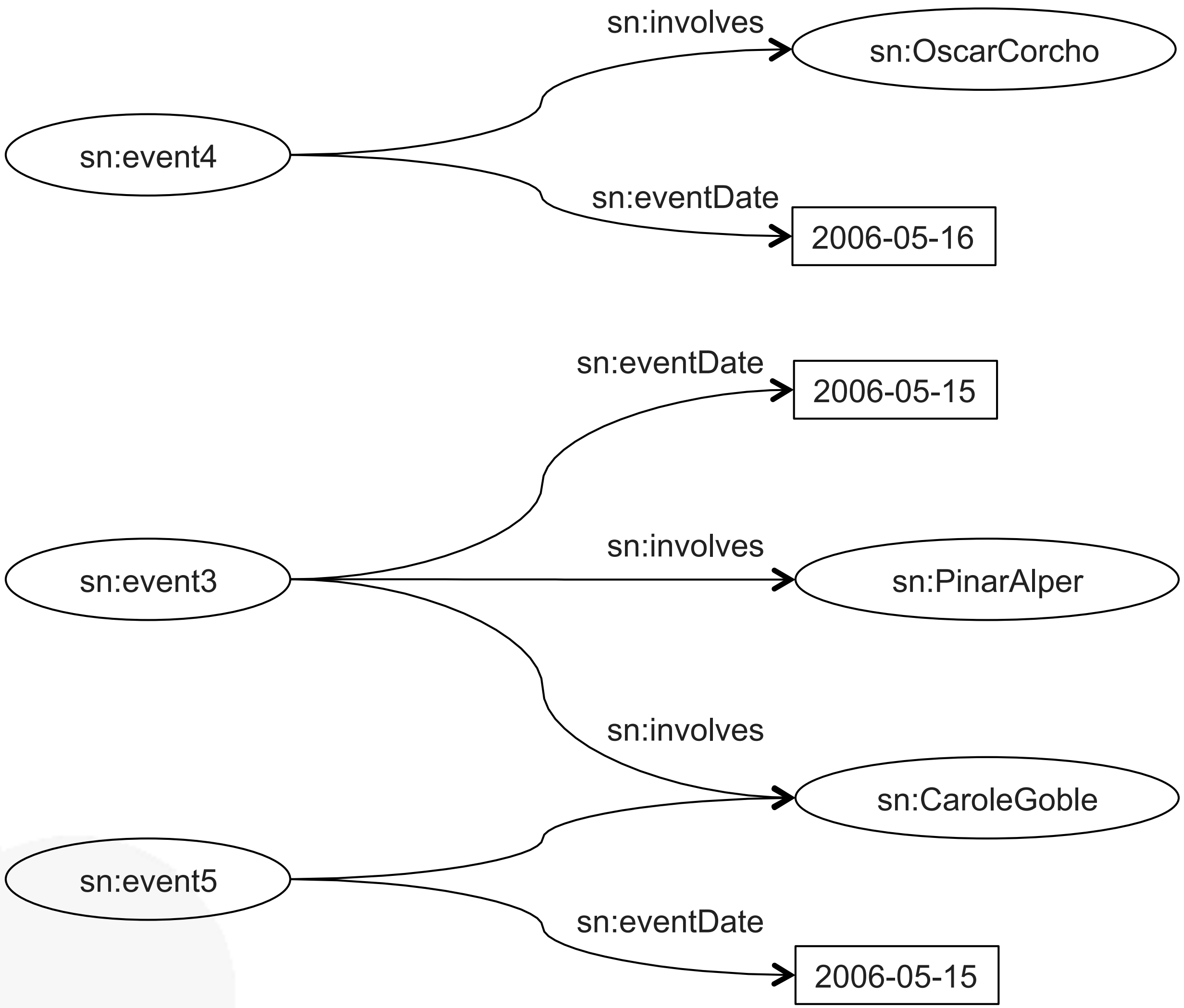
- Objective
  - Get used to the different syntaxes of RDF
- Tasks
  - Take the text of an RDF file and create its corresponding graph
  - Take an RDF graph and create its corresponding RDF/XML and N3 files

Exercise 1.a. Create a graph from a file



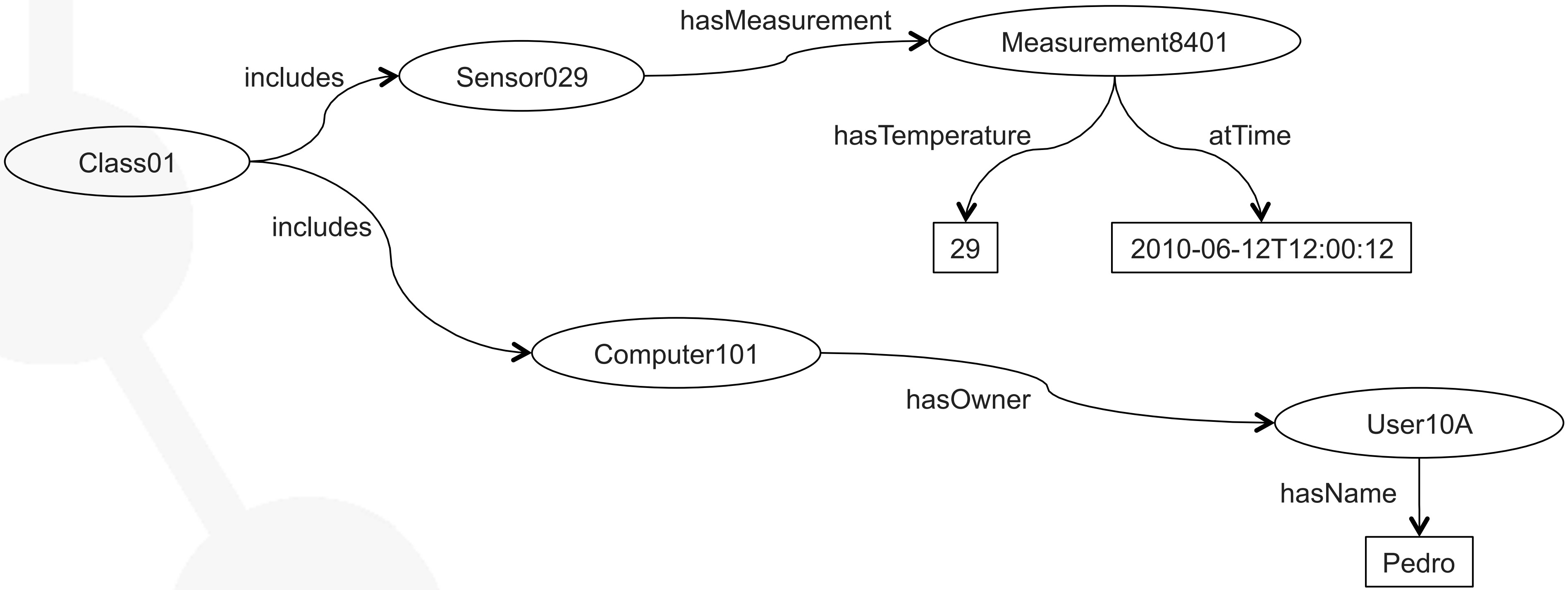
- Open the file StickyNote_PureRDF.rdf
- Create the corresponding graph from it
- Compare your graph with those of your colleagues

Exercise 1.a. StickyNote_PureRDF.rdf



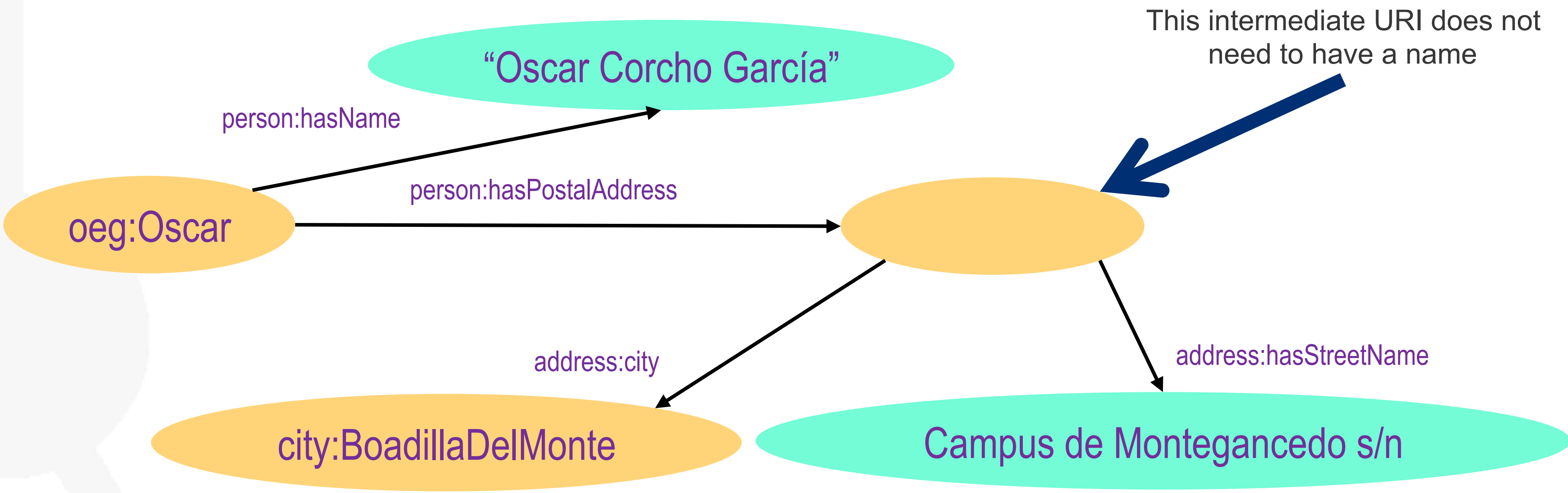
Exercise 1.b. Create files from a graph

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



Blank nodes: structured property values

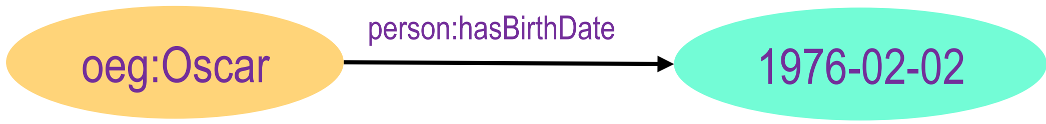
- Most real-world data involves structures that are more complicated than sets of RDF triple statements



- In RDF/XML, it is an <rdf:Description> node with no rdf:about
- In N3, it is a resource identifier that starts with '_'
  - E.g., "_:nodeX"

Typed literals

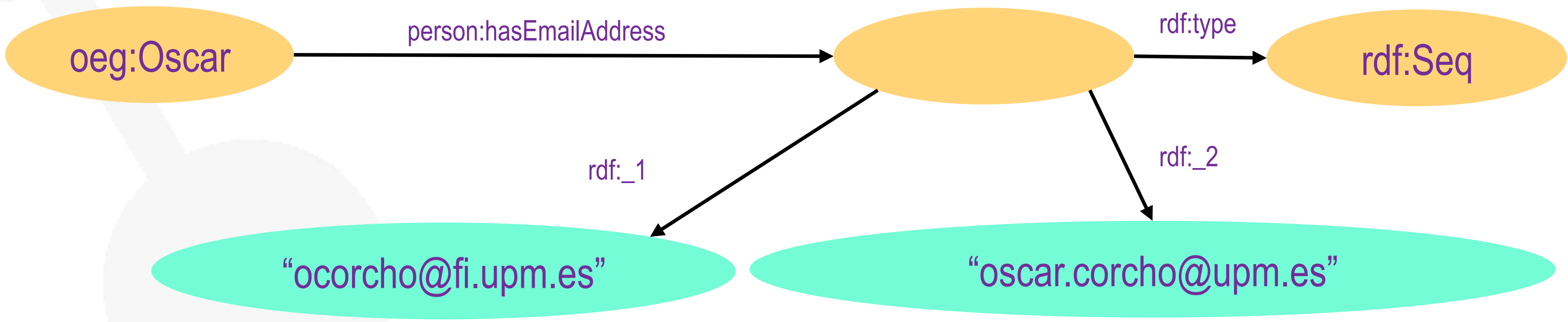
- So far, all values have been presented as strings
- XML Schema datatypes can be used to specify values (objects in some RDF triple statements)



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

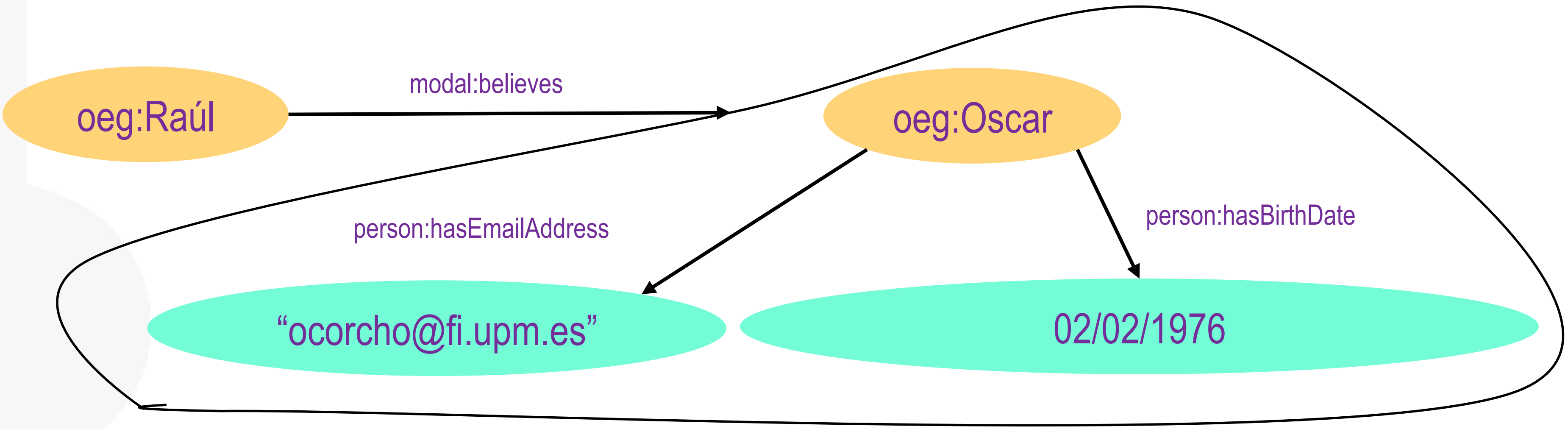
RDF Containers

- 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"

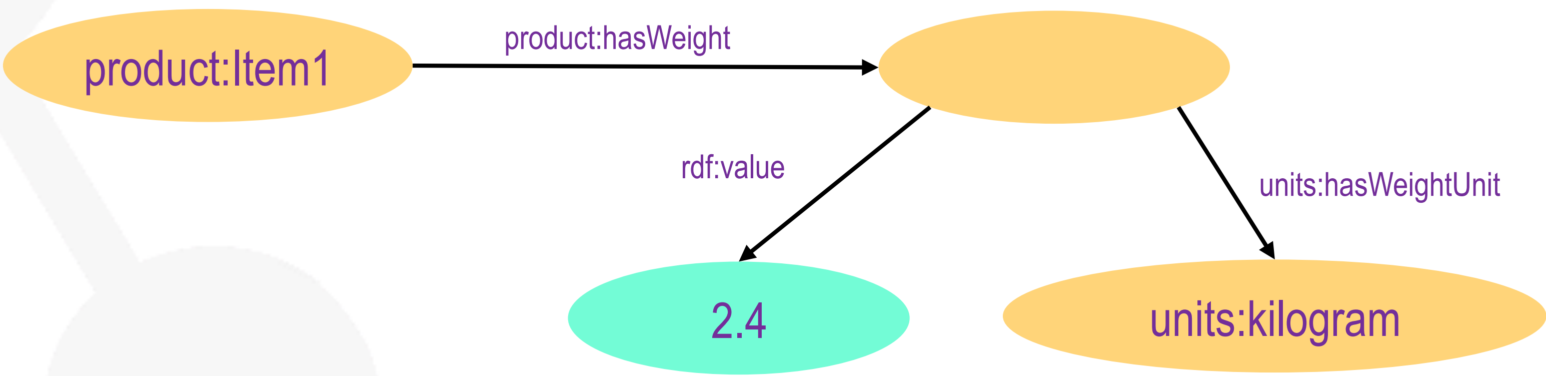


- RDF Reification
  - Allows expressing beliefs (and other modalities)
  - Allows expressing trust models, digital signatures, etc.
  - Allows expressing metadata about metadata

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 most important value is 2.4, which is expressed with rdf:value

- Scarcely used



Index

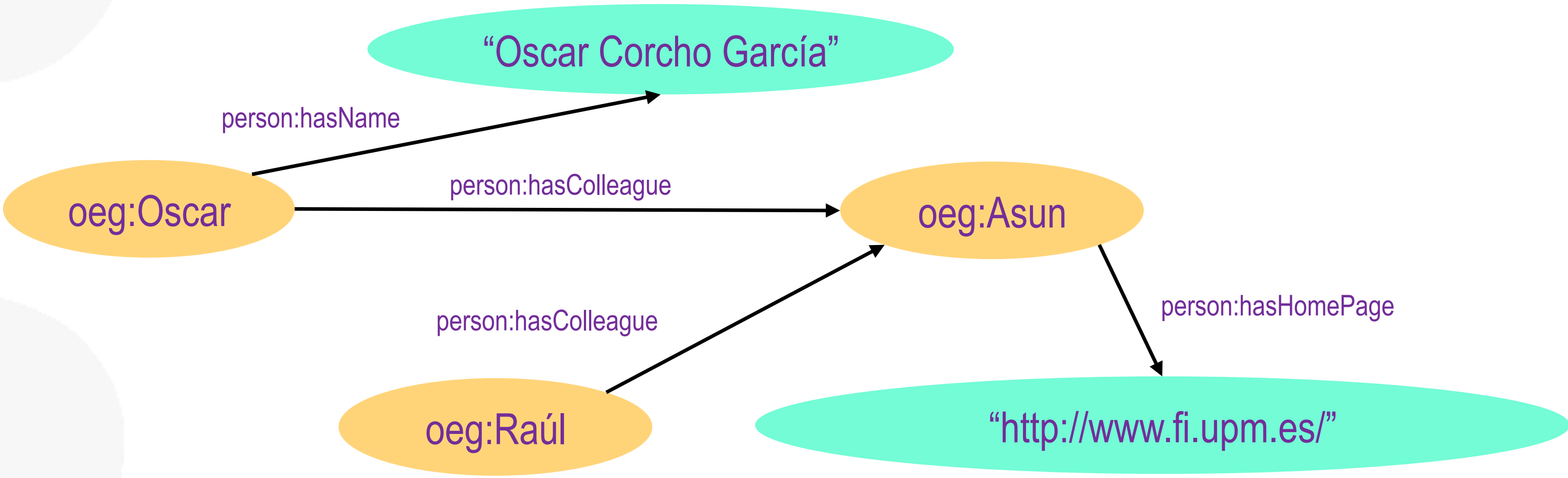
- Resource Description Framework (RDF)
  - RDF primitives
  - **Reasoning with RDF**
- RDF Schema
  - RDF Schema primitives
  - Reasoning with RDFS
- RDF(S) Management APIs
- Web Ontology Language (OWL)
  - OWL primitives
  - Reasoning with OWL

RDF inference. Graph matching techniques

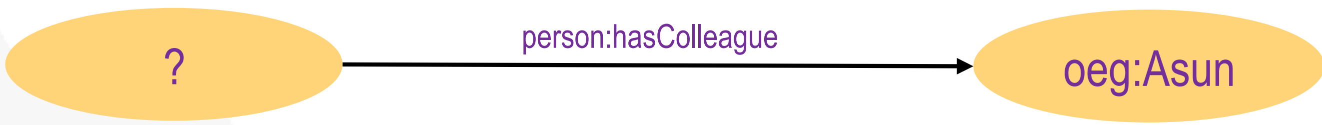
- RDF inference is based on graph matching techniques
- Basically, the RDF inference process consists of the following steps:
  - Transform an RDF query into a template graph that has to be matched against the RDF graph
    - It contains constant and variable nodes, and constant and variable edges between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



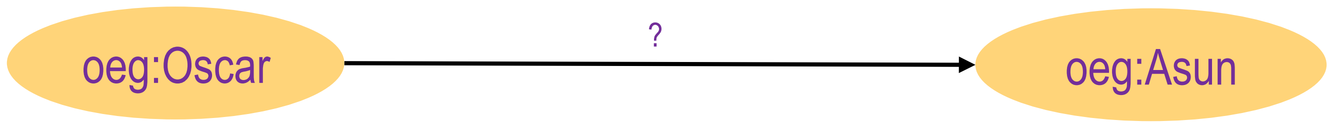
- **Query**: “Tell me who are the persons who have Asun as a colleague”



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

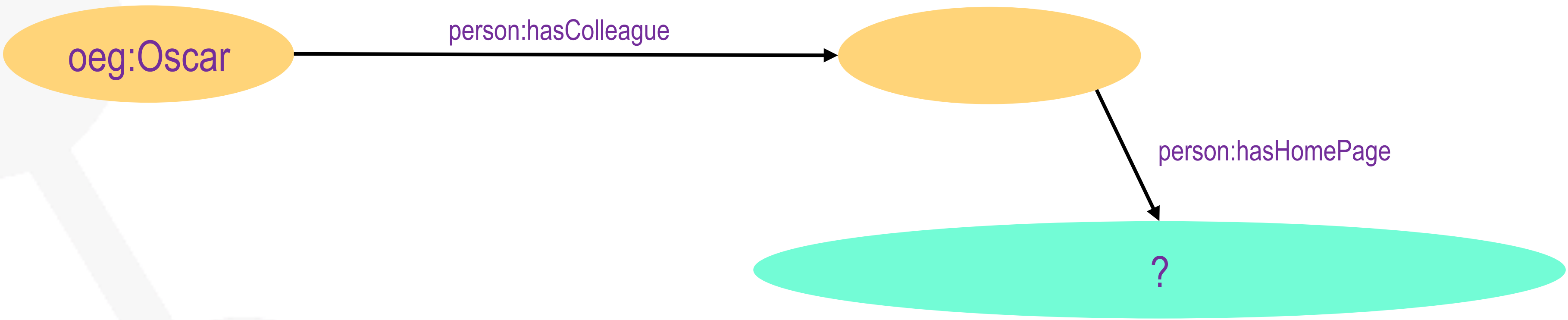
RDF inference. Examples (II)

- **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

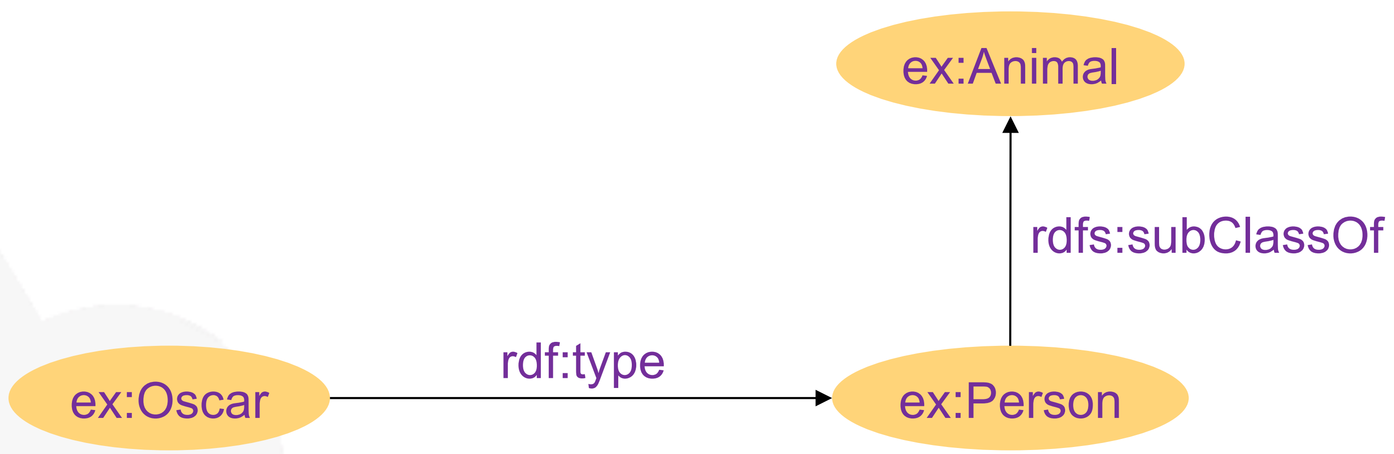
Rule Name	if E contains	then add
rdf1	uuu aaa yyy .	aaa `rdf:type rdf:Property .`
rdf2	uuu aaa lll . where lll is a well-typed XML literal .	_:nnn `rdf:type rdf:XMLLiteral .` where _:nnn identifies a blank node allocated to lll by rule lg.

Index

- Resource Description Framework (RDF)
  - RDF primitives
  - Reasoning with RDF
- **RDF Schema**
  - **RDF Schema primitives**
  - Reasoning with RDFS
- RDF(S) Management APIs
- Web Ontology Language (OWL)
  - OWL primitives
  - Reasoning with OWL

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
<?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

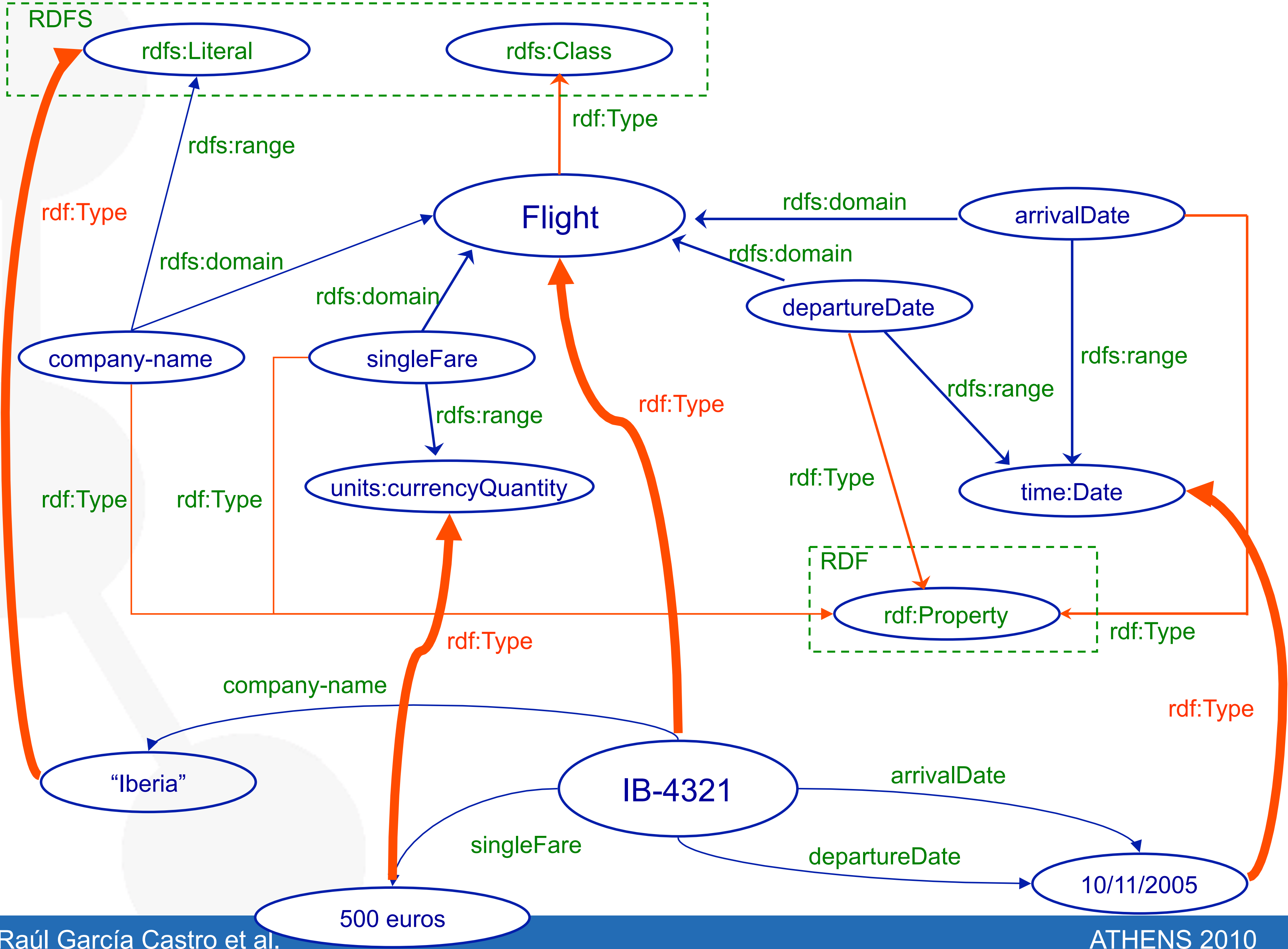
```
…
  <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasHomePage"/>
  <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasColleague">
    <rdfs:domain rdf:resource=" http://www.ontologies.org/ontologies/people#Person"/>
    <rdfs:range rdf:resource=" http://www.ontologies.org/ontologies/people#Person"/>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



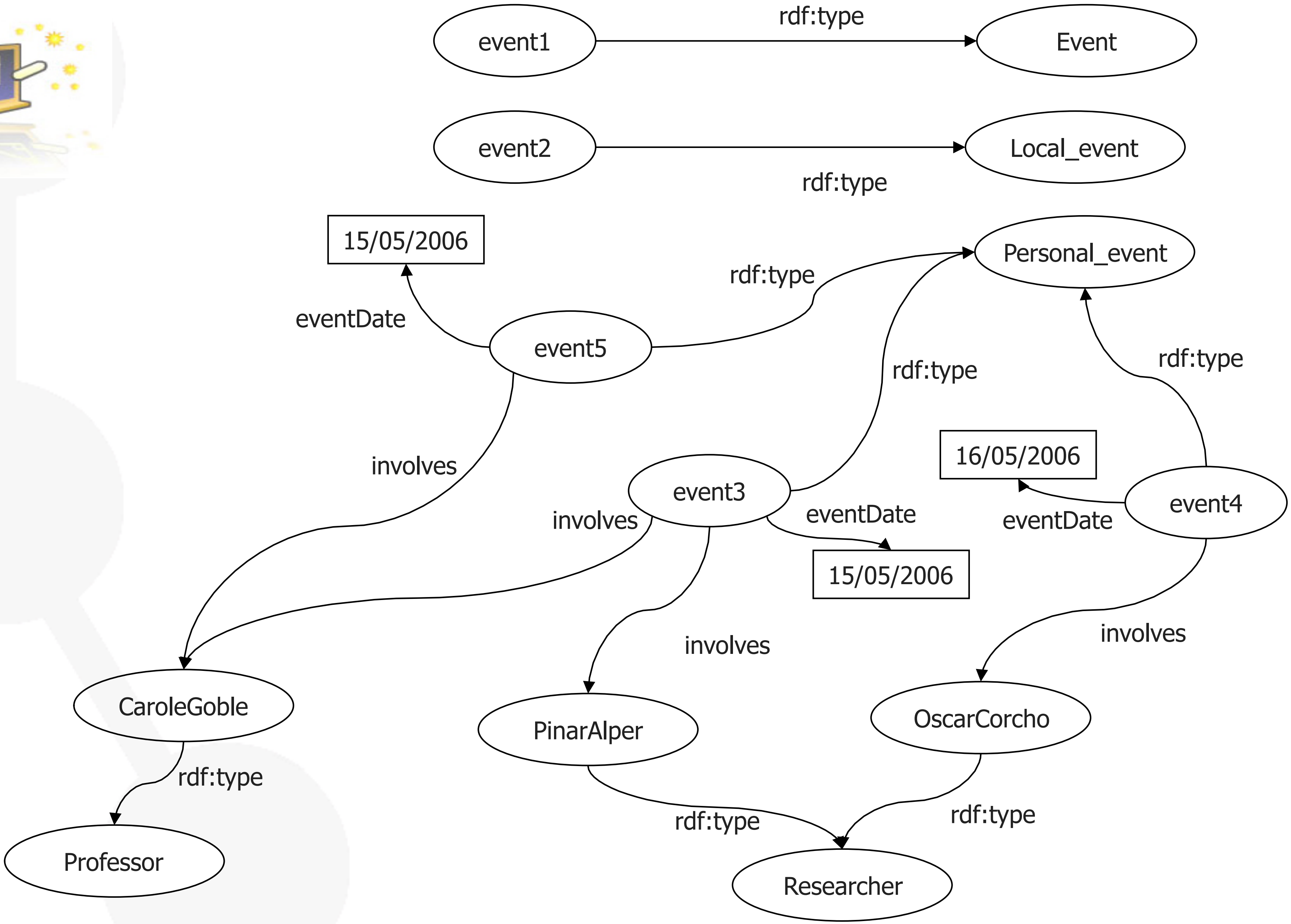
Exercise

- 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

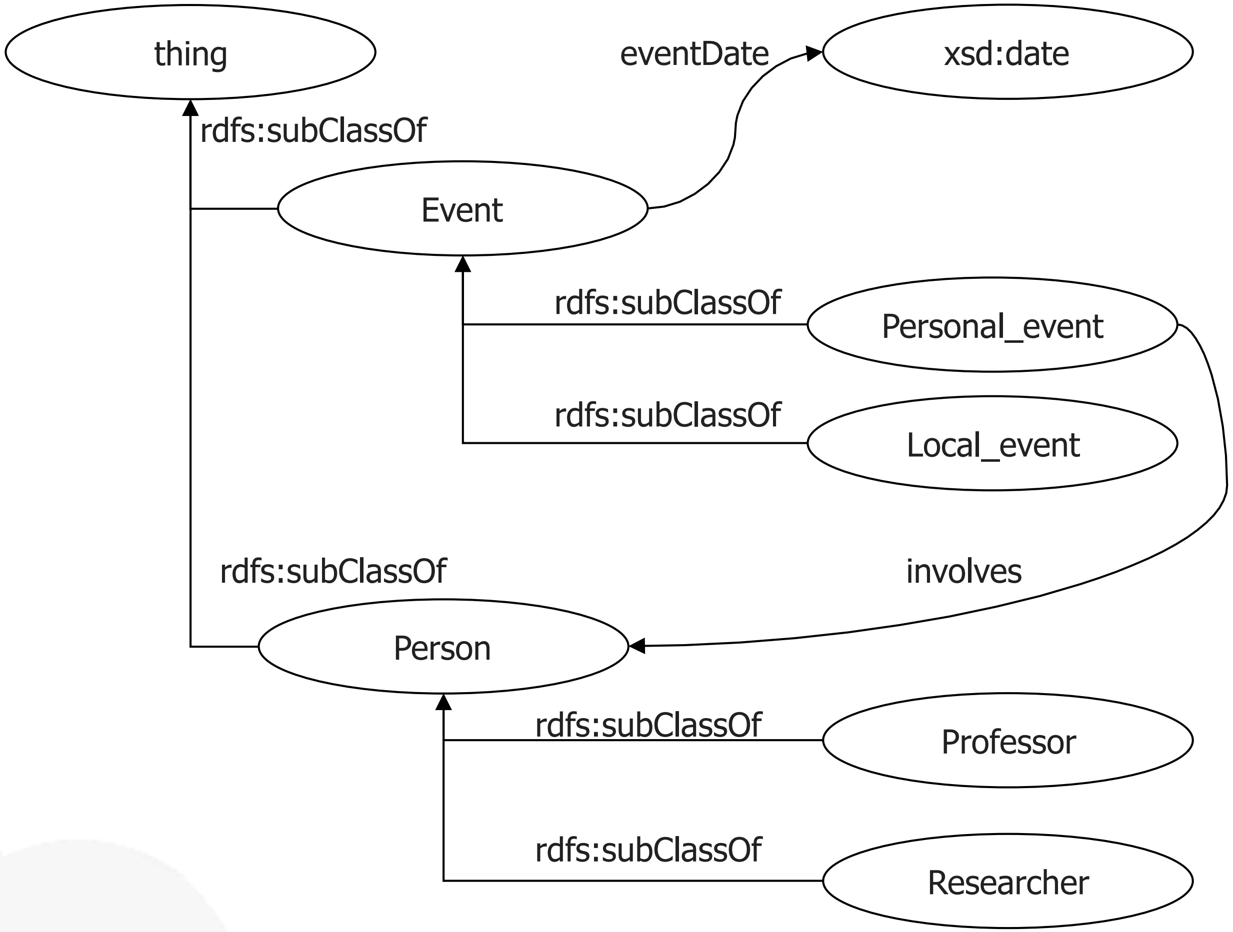
Exercise 2.a. Create a graph from a file

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

Exercise 2.a. StickyNote.rdf

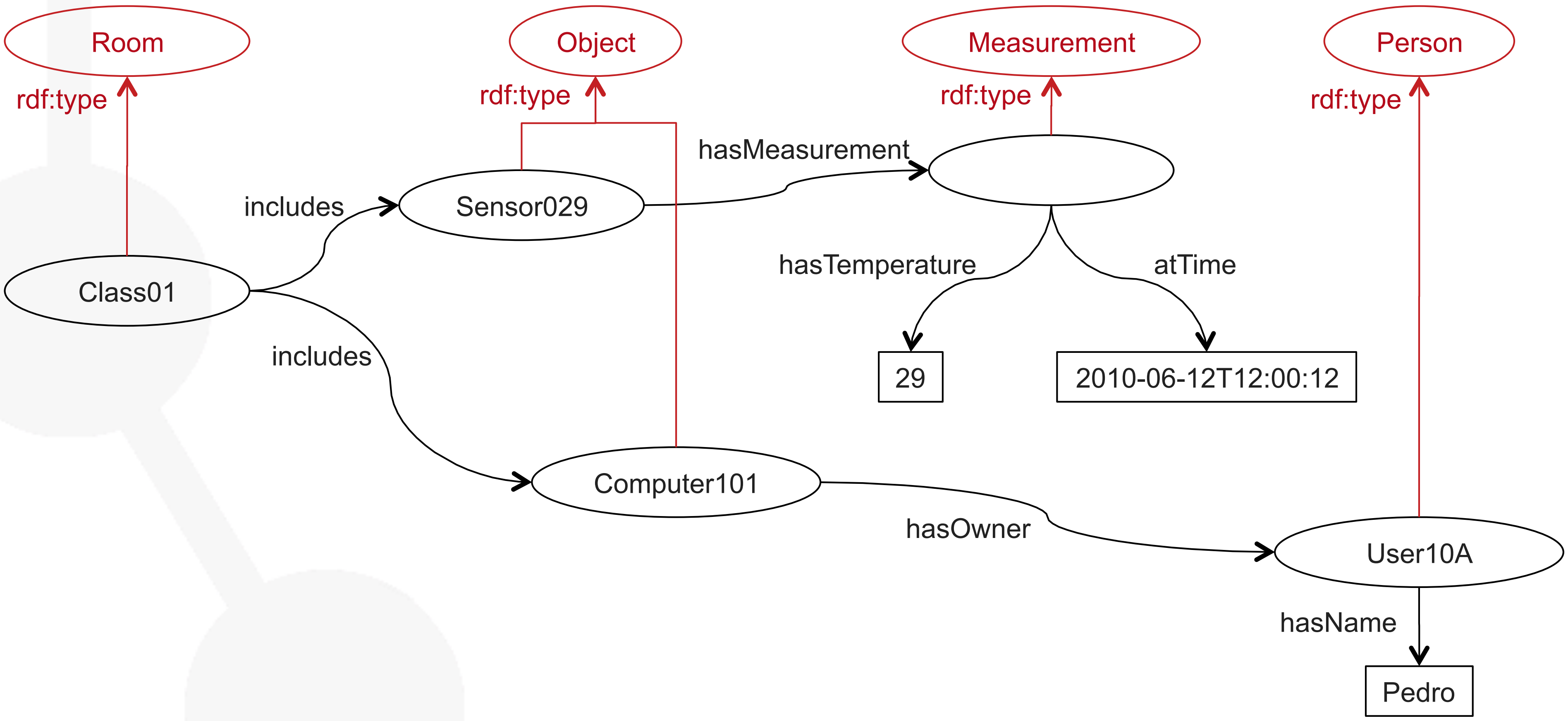


Exercise 2.a. StickyNote.rdfs



Exercise 2.b. Create files from a graph

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



Index

- Resource Description Framework (RDF)
  - RDF primitives
  - Reasoning with RDF
- RDF Schema
  - RDF Schema primitives
  - **Reasoning with RDFS**
- RDF(S) Management APIs
- Web Ontology Language (OWL)
  - OWL primitives
  - Reasoning with OWL

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

- Objective
  - Understand the features of RDF(S) for implementing ontologies, including its limitations
- Tasks
  - From your domain description, create the RDF(S) graph
    - First only include the vocabulary from the domain
    - Then include references to the RDF and RDFS vocabularies

Index

- Resource Description Framework (RDF)
  - RDF primitives
  - Reasoning with RDF
- RDF Schema
  - RDF Schema primitives
  - Reasoning with RDFS
- **RDF(S) Management APIs**
- Web Ontology Language (OWL)
  - OWL primitives
  - Reasoning with OWL

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 http://esw.w3.org/topic/SemanticWebTools
- 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/

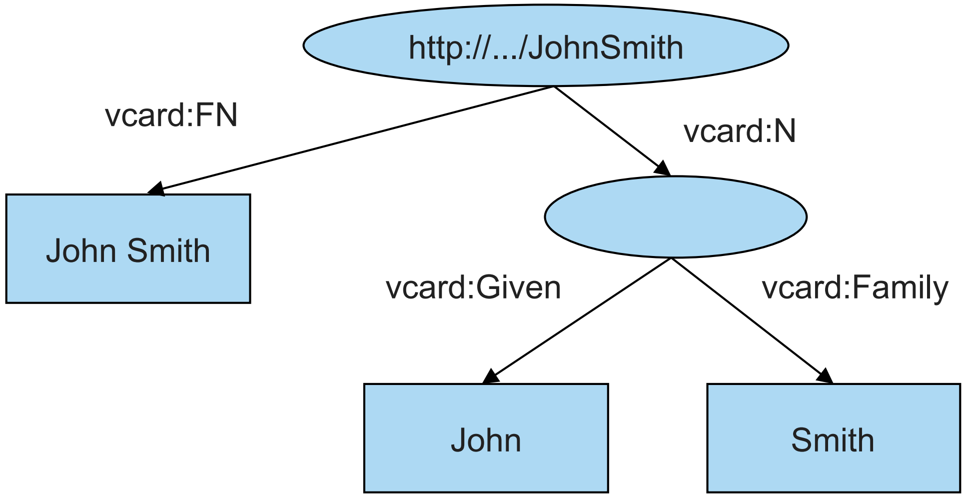
Jena

- 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/index.html
- 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/

Index

- Resource Description Framework (RDF)
  - RDF primitives
  - Reasoning with RDF
- RDF Schema
  - RDF Schema primitives
  - Reasoning with RDFS
- RDF(S) Management APIs
- **Web Ontology Language (OWL)**
  - **OWL primitives**
  - Reasoning with OWL

Description Logics

- A family of logic-based Knowledge Representation formalisms
  - Descendants of semantic networks and KL-ONE
  - Describe domain in terms of concepts (classes), roles (relationships) and individuals
    - Specific languages characterised by the constructors and axioms used to assert knowledge about classes, roles and individuals.
    - Example: ALC (the least expressive language in DL that is propositionally closed)
      - Constructors: boolean (and, or, not)
      - Role restrictions

- Distinguished by:
  - Model theoretic semantics
    - Decidable fragments of FOL
    - Closely related to Propositional Modal & Dynamic Logics
  - Provision of inference services
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimised)

Structure of DL Ontologies

- A DL ontology can be divided into two parts:
  - **Tbox** (Terminological KB): a set of axioms that describe the structure of a domain:
    - Doctor ⊆ Person
    - Person ⊆ Man ∪ Woman
    - HappyFather ⊆ Man ∩ ∀hasDescendant.(Doctor ∪ ∀hasDescendant.Doctor)
  - **Abox** (Assertional KB): a set of axioms that describe a specific situation:
    - John ∈ HappyFather
    - hasDescendant (John, Mary)

OWL (1)

February 2004

Web Ontology Language

Built on top of RDF(S)

Three layers:

- OWL Lite
  - A small subset of primitives
  - Easier for frame-based tools to transition to
- OWL DL
  - Description logic
  - Decidable reasoning
- OWL Full
  - RDF extension, allows metaclasses

Several syntaxes:

- Abstract syntax
- Manchester syntax
- RDF/XML

OWL 2 (I). New features

- October 2009

- New features
  - Syntactic sugar
    - Disjoint union of classes
  - New expressivity
    - Keys
    - Property chains
    - Richer datatypes, data ranges
    - Qualified cardinality restrictions
    - Asymmetric, reflexive, and disjoint properties
    - Enhanced annotation capabilities

- New syntax
  - OWL2 Manchester syntax

OWL 2 (II). Three new profiles

- OWL2 EL
  - Ontologies that define very large numbers of classes and/or properties
  - Ontology consistency, class expression subsumption, and instance checking can be decided in polynomial time
- OWL2 QL
  - Sound and complete query answering is in LOGSPACE (more precisely, in AC^0) with respect to the size of the data (assertions)
  - Provides many of the main features necessary to express conceptual models (UML class diagrams and ER diagrams)
  - It contains the intersection of RDFS and OWL 2 DL
- OWL2 RL
  - Inspired by Description Logic Programs and pD*
  - Syntactic subset of OWL 2 which is amenable to implementation using rule-based technologies
  - Scalable reasoning without sacrificing too much expressive power
  - Designed for
    - OWL applications trading the full expressivity of the language for efficiency,
    - RDF(S) applications that need some added expressivity from OWL 2.

OWL: Most common constructors

Intersection:	$C_1 \cap \ldots \cap C_n$	intersectionOf	Human ∩ Male
Union:	$C_1 \cup \ldots \cup C_n$	unionOf	Doctor ∪ Lawyer
Negation:	$\neg C$	complementOf	¬Male
Nominals:	$\{x_1\} \cup \ldots \cup \{x_n\}$	oneOf	{john} ∪ ... ∪ {mary}
Universal restriction:	$\forall P.C$	allValuesFrom	∀hasChild.Doctor
Existential restriction:	$\exists P.C$	someValuesFrom	∃hasChild.Lawyer
Maximum cardinality:	$\leq nP[.C]$	maxCardinality (qualified or not)	≤3hasChild[.Doctor]
Minimum cardinality:	$\geq nP[.C]$	minCardinality (qualified or not)	≥1hasChild[.Male]
Exact cardinality:	$=nP[.C]$	exactCardinality (qualified or not)	=1hasMother[.Female]
Specific Value:	$\exists P.\{x\}$	hasValue	∃hasColleague.{Matthew}
Local reflexivity:	--	hasSelf	Narcisist ≡ Person ∩ hasSelf(loves)
Keys	--	hasKey	hasKey(Person, passportNumber, country)
Subclass	$C1 \subseteq C2$	subClassOf	Human ⊆ Animal ∩ Biped
Equivalence	$C1 \equiv C2$	equivalentClass	Man ≡ Human ∩ Male
Disjointness	$C1 \cap C2 \subseteq \bot$	disjointWith, AllDisjointClasses	Male ∩ Female ⊆ ⊥
DisjointUnion	$C \equiv C_1 \cup \ldots \cup C_n$ and $C_i \cap C_j \subseteq \bot$ forall $i \neq j$	disjointUnionOf	Person DisjointUnionOf (Man, Woman)

Metaclasses and annotations on axioms are also valid in OWL2, and declarations of classes have to provided.

Full list available in reference specs and in the Quick Reference Guide: http://www.w3.org/2007/OWL/refcard

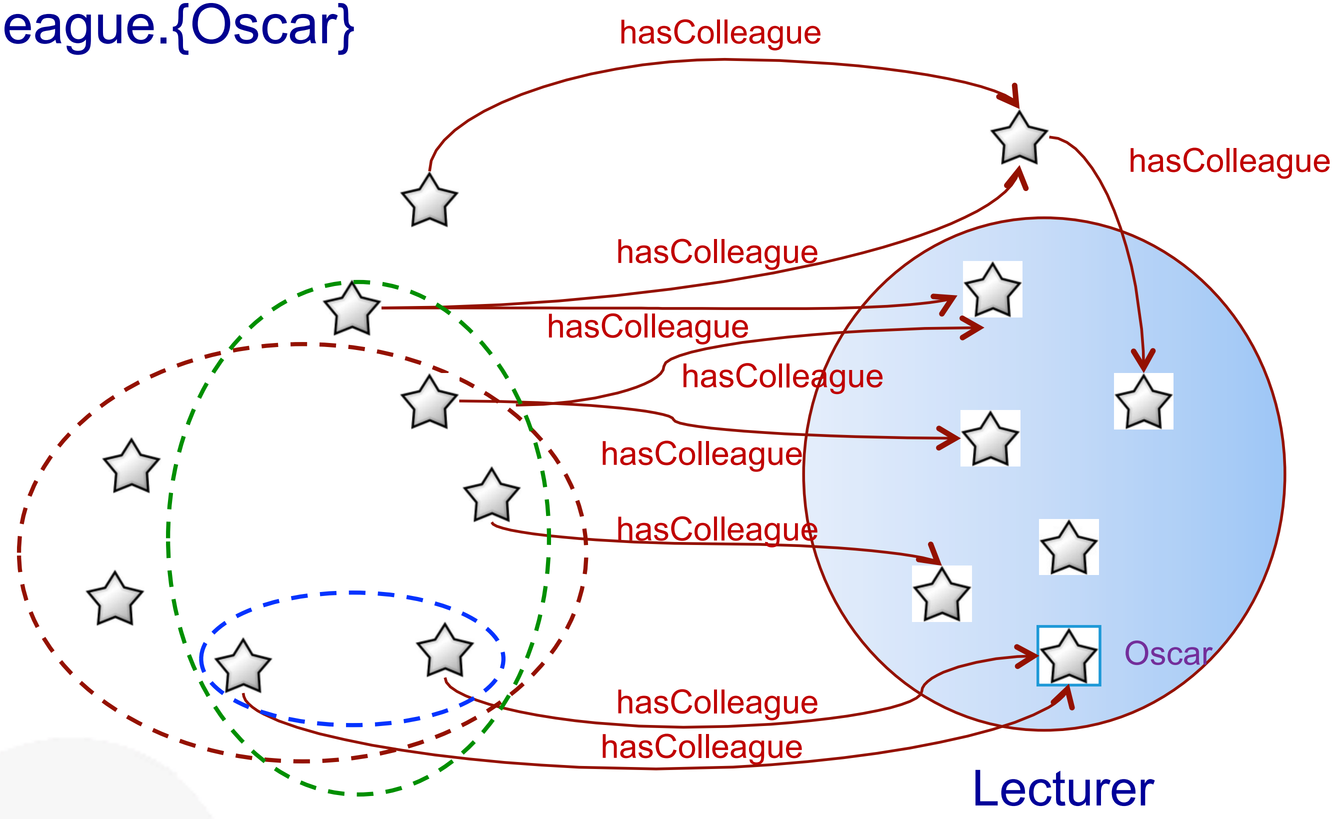
OWL: Most common constructors

Subproperty	$P1 \subseteq P2$	subPropertyOf	$hasDaughter \subseteq hasChild$
Equivalence	$P1 \equiv P2$	equivalentProperty	$cost \equiv price$
DisjointProperties	$P1 \cap ... \cap Pn \subseteq \perp$	disjointObjectProperties	$hasDaughter \cap hasSon \subseteq \perp$
Inverse	$P1 \equiv P2^-$	inverseOf	$hasChild \equiv hasParent^-$
Transitive	$P^+ \subseteq P$	TransitiveProperty	$ancestor^+ \subseteq ancestor$
Functional	$T \subseteq \leq 1P$	FunctionalProperty	$T \subseteq \leq 1hasMother$
InverseFunctional	$T \subseteq \leq 1P^-$	InverseFunctionalProperty	$T \subseteq \leq 1hasPassportID^-$
Reflexive		ReflexiveProperty	
Irreflexive		IrreflexiveProperty	
Asymmetric		AsymmetricProperty	
Property chains	$P \equiv P1 \circ ... \circ Pn$	propertyChainAxiom	$hasUncle \subseteq hasFather \circ hasBrother$
Equivalence	$\{x1\} \equiv \{x2\}$	sameIndividualAs	{oeg:OscarCorcho}≡{img:Oscar}
Different	$\{x1\} \equiv \neg\{x2\}$	differentFrom, AllDifferent	$\{john\} \equiv \neg\{peter\}$
NegativePropertyAssertion		NegativeDataPropertyAssertion	¬{hasAge john 35}
		NegativeObjectPropertyAssertion	¬{hasChild john peter}

Besides, top and bottom object and datatype properties exist

Do we understand these constructors?

- ∃hasColleague.Lecturer
- ∀hasColleague.Lecturer
- ∃hasColleague.{Oscar}



- Resource Description Framework (RDF)
  - RDF primitives
  - Reasoning with RDF
- RDF Schema
  - RDF Schema primitives
  - Reasoning with RDFS
- **RDF(S) Management APIs**
- **Web Ontology Language (OWL)**
  - OWL primitives
  - **Reasoning with OWL**

Basic Inference Tasks

- Subsumption – check knowledge is correct (captures intuitions)
  - Does C subsume D w.r.t. ontology O? (in *every* model I of O, $C^I \subseteq D^I$)
- Equivalence – check knowledge is minimally redundant (no unintended synonyms)
  - Is C equivalent to D w.r.t. O? (in *every* model I of O, $C^I = D^I$)
- Consistency – check knowledge is meaningful (classes can have instances)
  - Is C satisfiable w.r.t. O? (there exists *some* model I of O s.t. $C^I \neq \emptyset$)
- Instantiation and querying
  - Is x an instance of C w.r.t. O? (in *every* model I of O, $x^I \in C^I$)
  - Is (x,y) an instance of R w.r.t. O? (in *every* model I of O, $(x^I,y^I) \in R^I$)
- All reducible to KB satisfiability or concept satisfiability w.r.t. a KB
- Can be decided using highly optimised tableaux reasoners

Interesting results (I). Automatic classification

And old lady is a person who is elderly and female.
Old ladies must have some animal as pets and all their pets are cats.

$$elderly \subseteq person \cap adult$$

$$woman \equiv person \cap female \cap adult$$

$$catOwner \equiv person \cap \exists hasPet.cat$$

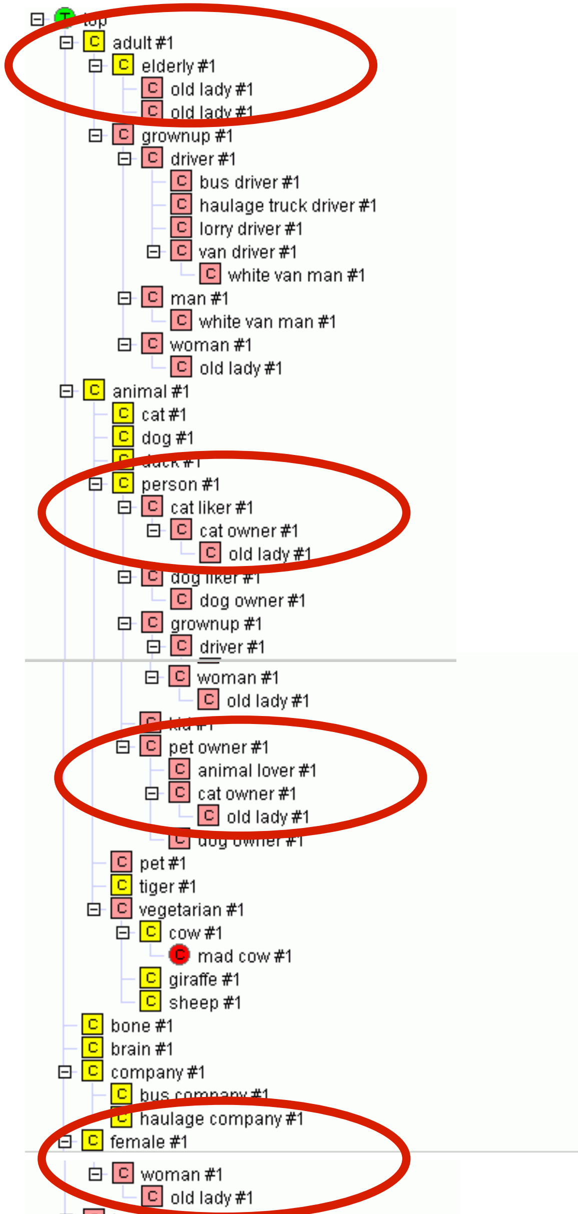
$$oldLady \equiv person \cap female \cap elderly$$

$$oldLady \subseteq \exists hasPet.animal \cap \forall hasPet.cat$$

We obtain:

Old ladies must be women.
Every old lady must have a pet cat
Hence, every old lady must be a cat owner

$$oldLady \subseteq woman \cap elderly \cap catOwner$$



Interesting results (II). Instance classification

A pet owner is a person who has animal pets
Old ladies must have some animal as pets and all their pets are cats.
Has pet has domain person and range animal
Minnie is a female, elderly, who has a pet called Tom.

$$petOwner \equiv person \cap \exists hasPet.animal$$

$$oldLady \subseteq \exists hasPet.animal \cap \forall hasPet.cat$$

$$hasPet \subseteq (person, animal)$$

$$Minnie \in female \cap elderly$$

$$hasPet(Minnie, Tom)$$

We obtain:

- Minnie is a person
- Hence, Minnie is an old lady
- Hence, Tom is a cat

$$Minnie \in person, Tom \in animal$$

$$Minnie \in petOwner$$

$$Minnie \in oldLady$$

$$Tom \in cat$$

Interesting results (III). Instance classification and redundancy detection

An animal lover is a person who has at least three pets
Walt is a person who has pets called Huey, Louie and Dewey.

$$animalLover \equiv person \cap (\geq 3 hasPet)$$

$$Walt \in person$$

$$hasPet(Walt, Huey)$$

$$hasPet(Walt, Louie)$$

$$hasPet(Walt, Dewey)$$

We obtain:

- Walt is an animal lover
- Walt is a person is redundant

$$Walt \in animalLover$$

Interesting results (IV). Instance classification

A van is a type of vehicle
A driver must be adult
A driver is a person who drives vehicles
A white van man is a man who drives vans and white things
White van mans must read only tabloids
Q123ABC is a white thing and a van
Mick is a male who reads Daily Mirror and drives Q123ABC

$$van \subseteq vehicle$$
$$driver \subseteq adult$$
$$driver \equiv person \cap \exists drives.vehicle$$
$$whiteVanMan \equiv man \cap \exists drives.(van \cap whiteThing)$$
$$whiteVanMan \subseteq \forall reads.tabloid$$
$$Q123ABC \in whiteThing \cap van$$
$$Mick \in male$$
$$reads(Mick, DailyMirror)$$
$$drives(Mick, Q123ABC)$$

We obtain:

- Mick is an adult
- Mick is a white van man
- Daily Mirror is a tabloid

$$Mick \in adult$$
$$Mick \in whiteVanMan$$
$$DailyMirror \in tabloid$$

Interesting results (V). Consistency checking

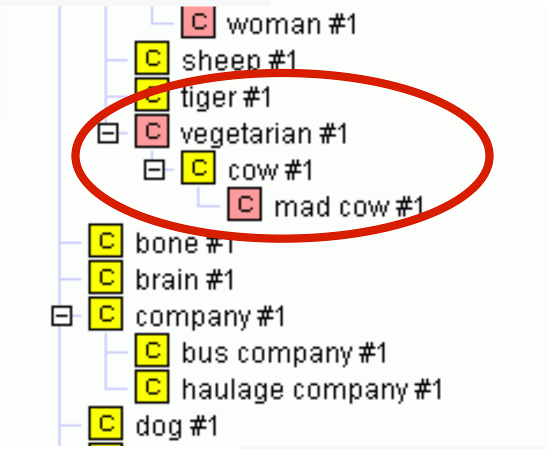
Cows are vegetarian.
A vegetarian is an animal that does not eat animals nor parts of animals.
A mad cow is a cow that eats brains that can be part of a sheep

$$cow \subseteq vegetarian$$

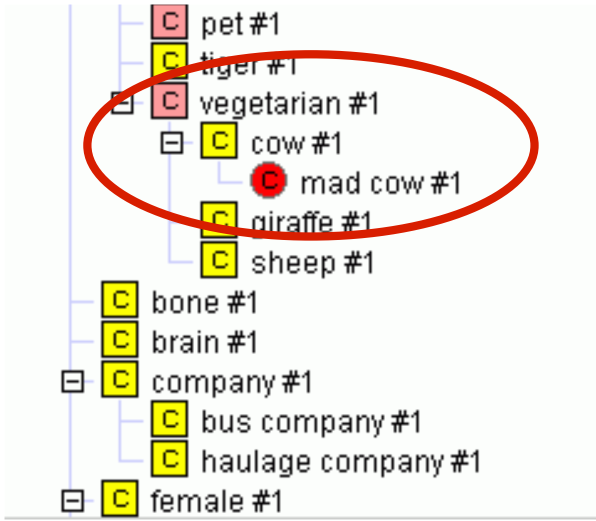
$$vegetarian \equiv animal \cap \forall eats.\neg animal \cap \forall eats.\neg(\exists partOf.animal))$$

$$madCow \equiv cow \cap \exists eats.(brain \cup \exists partOf.sheep)$$

$$(animal \cup \exists partOf.animal) \cap (plant \cup \exists partOf.plant) \subseteq \perp$$



We obtain:
Mad cow is unsatisfiable



OWL Classifier limitations

- Numbers and strings
  - Simple concrete data types in spec
  - User defined XML data types enmeshed in standards disputes
  - No standard classifier deals with numeric ranges
    - Although several experimental ones do

- is-part-of and has-part
  - Totally doubly-linked structures scale horridly

- Handling of individuals
  - Variable with different classifiers
  - oneOf works badly with all classifiers at the moment

Exercise

- Objective
  - Understand the features of OWL for implementing ontologies
- Tasks
  - Take your RDF(S) model and convert it into OWL
    - What new things can you express in the model?

Main References

- Dean M, Schreiber G (2004) OWL Web Ontology Language Reference. W3C Recommendation.
  http://www.w3.org/TR/owl-ref/
- W3C OWL Working Group
  http://www.w3.org/2007/OWL/wiki/OWL_Working_Group

Thank you for your attention!

Speaker: Raúl García Castro
rgarcia@fi.upm.es

ATHENS 2010
Madrid, Spain