Semantic Days 2012 Tutorial
Semantic Web Technologies
Lecture 4: OWL, the Web Ontology Language

Martin Giese

8th May 2012

Department of Informatics

University of Oslo

Outline

Modeling

- Remember what we said about modeling in the intro. . .
  - Fix a vocabulary
  - Describe interaction of terms in the vocabulary
- So far, this has been like W3C-ized database technology
  - Triples instead of tables
  - SPARQL instead of SQL
- Now, we add a modeling language
  - Similar to a database schema
  - Even more similar to UML class diagrams
  - But with functionality beyond both
  - Useful functionality to *align vocabularies*

The RDFS vocabulary

- RDFS adds the concept of "classes" which are like *types* or *sets* of resources
- A predefined vocabulary allows statements about classes
- Defined resources:
  - `rdfs:Resource`: The class of resources, everything.
  - `rdfs:Class`: The class of classes.
  - `rdf:Property`: The class of properties (from `rdf`)
- Defined properties:
  - `rdf:type`: relate resources to classes they are members of
  - `rdfs:domain`: The domain of a relation.
  - `rdfs:range`: The range of a relation.
  - `rdfs:subClassOf`: Concept inclusion.
  - `rdfs:subPropertyOf`: Property inclusion.

Semantics

- The meaning of this vocabulary is given through *inference rules*
- Given some statements, a rule allows to infer other statements
- Totally some 20 rules for RDFS
- E.g. rule RDFS9:

$$\frac{\begin{array}{c} C \texttt{ rdfs:subClassOf } D \\ x \texttt{ rdf:type } C \end{array}}{x \texttt{ rdf:type } D}$$

- Example of inference:

$$\frac{\begin{array}{c} \texttt{:City rdfs:subClassOf :Place} \\ \texttt{:Oslo rdf:type :City} \end{array}}{\texttt{:Oslo rdf:type :Place}}$$

- Asserted and inferred triples

Rules for Properties

- Sub-properties:

$$\frac{R \texttt{ rdfs:subPropertyOf } S \qquad x\,R\,y}{x\,S\,y}$$

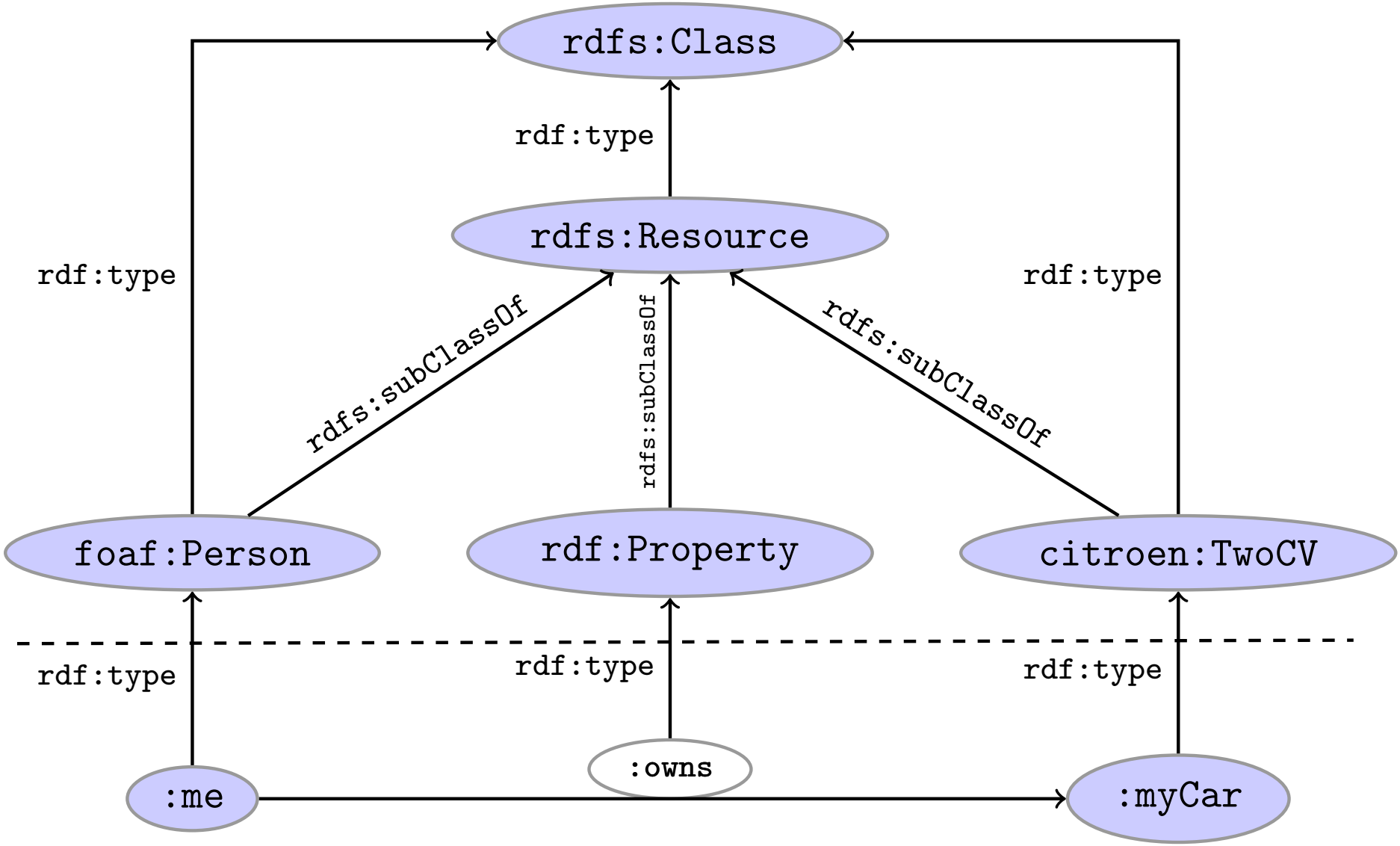
- Domain:

$$\frac{R \texttt{ rdfs:domain } C \qquad x\,R\,y}{x \texttt{ rdf:type } C}$$

- Range:

$$\frac{R \texttt{ rdfs:range } C \qquad x\,R\,y}{y \texttt{ rdf:type } C}$$

Example


rdfs:Class
rdf:type
rdfs:Resource
rdf:type
rdf:type
rdfs:subClassOf
rdfs:subClassOf
rdfs:subClassOf
foaf:Person
rdf:Property
citroen:TwoCV
rdf:type
rdf:type
rdf:type
:owns
:me
:myCar

Outline

OWL is simpler...

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- "Data level" with "instances"
- "Ontology level" with properties and "classes"
- Classes and properties *not* part of the domain!
- Can have `rdf:type` relation between data objects and classes
- Properties connect instances
- Allow a fixed vocabulary for relations between classes and properties
- Interpret:
  - Class as set of data objects
  - Property as relation between data objects
- A setting well-studied as *Description Logics*

OWL Quick Facts

OWL:

- Acronym for *The Web Ontology Language*.
- Became a W3C recommendation in 2004.
- The undisputed standard ontology language.
- Superseded by OWL 2 in 2009;
  - a backwards compatible extension that adds new capabilities.
- OWL is a language to express "ontologies"
- i.e. express facts about a domain, like RDFS
- Built on Description Logics, separation of data and ontology
- Combines DL expressiveness with RDF technology (URIs, namespaces, etc.)
- Extends RDFS with boolean operations, universal/existential restrictions, etc.

OWL Syntaxes

- Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- Same for OWL:
- Defined as a set of things that can be said about classes, properties, instances
- OWL/RDF: Uses RDF to express OWL ontologies
  - Then use any of the RDF serializations
- OWL/XML: a non-RDF XML format
- Functional OWL syntax: simple, used in definition
- Manchester OWL syntax: textual format used in some tools

OWL Concept Descriptions

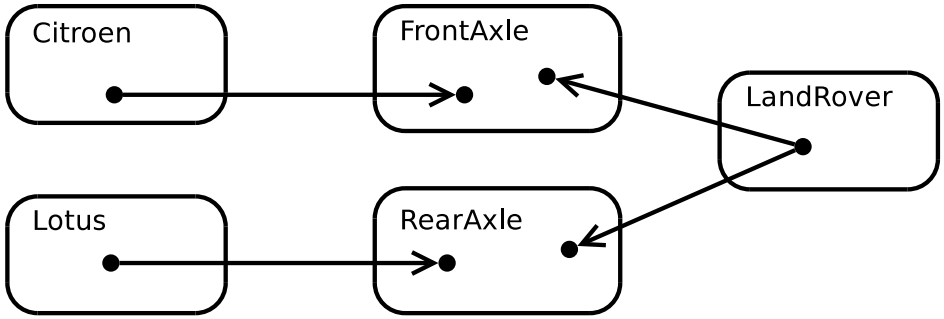
- In OWL, classes can be combined to form new concepts
- Boolean combinations:
  - (foaf:Person and not gender:Female) or rodents:Mouse
  - All things which are either persons and not female or mice
- Existential restrictions:
  - foaf:knows some gender:Female
  - All things which know some (at least one) female being
- Universal restrictions:
  - foaf:knows only gender:Female
  - All things which know *only* female beings
- And a few more.

OWL Axioms

- Concept descriptions don't say anything.
- There are essentially two kinds of things you can say in OWL:
- Concept membership
  - A resource belongs to a concept (`rdf:type`)
  - `:martingi rdf:type` (foaf:knows some gender:Female).
- Concept subsumption
  - Everything in one concept belongs to another concept (`rdfs:subClassOf`)
  - `gender:Male rdfs:subClassOf` (not gender:Female)
- Subsumption both ways: equivalence
  - Can be used to *define* terms
  - `:Woman owl:equivalentClass` (foaf:Person and gender:Female)

Example: Cars

- Assume:
  - All *Citroen* cars have one drive axle and that is the front axle
  - All *Lotus* cars have one drive axle and that is the rear axle
  - All *LandRover* cars have two drive axles, one front and one back



- Then the following axioms hold:
  - `:Citroen rdfs:subClassOf` (:driveAxle only :FrontAxle)
  - `:Lotus rdfs:subClassOf` (:driveAxle only :RearAxle)
  - `:LandRover rdfs:subClassOf`
    (:driveAxle some :FrontAxle and :driveAxle some :RearAxle)

Object Properties vs. Datatype Properties

- Can use RDF properties with resource or literal objects:
  - `:martin foaf:knows :simone .`
  - `:martin foaf:knows "Simone" .`
  - not the intended use, but OK for RDF.
- In OWL, every property has to be declared as one of

  object property only resources as objects
  datatype property only literals as objects
- Note: objects of datatype properties don't need to be typed
- Mostly a historical accident
- Doesn't hurt

Demo: Using Protégé

- Create a Car class
- Create an Axle class
- Create FrontAxle and RearAxle as subclasses
- Make the axle classes disjoint
- Add a driveAxle object property
- Add domain Car and range Axle
- Add 2CV, subclass of Car
- Add superclass driveAxle only FrontAxle
- Add Lotus, subclass of Car
- Add superclass driveAxle only RearAxle
- Add LandRover, subclass of Car
- Add superclass driveAxle some FrontAxle
- Add superclass driveAxle some RearAxle
- Add 4WD as subclass of Thing
- Make equivalent to driveAxle some RearAxle and driveAxle some FrontAxle
- Classify.
- Show inferred class hierarchy: Car $\sqsupseteq$ 4WD $\sqsupseteq$ LandRover
- Tell story of 2CV Sahara, which is a 2CV with two motors, one front, one back
- Add Sahara as subclass of 2CV
- Add 4WD as superclass of Sahara
- Classify.
- Show that Sahara is equivalent to bottom.
- Explain why. In particular, disjointness of front and rear axles

Protégé Recap

- Almost like using an OO modeling tool
- Remember: In the end it's
  - OWL concept descriptions
  - `rdf:type`
  - `rdfs:subClassOf`
- Many ways of saying things in OWL, more in Protégé