INF3580 – Semantic Technologies – Spring 2010

Lecture 8: OWL, the Web Ontology Language

Martin Giese

16th March 2010

Department of Informatics

University of Oslo

Today's Plan

1. Reminder: RDFS
2. Description Logics
3. Introduction to OWL

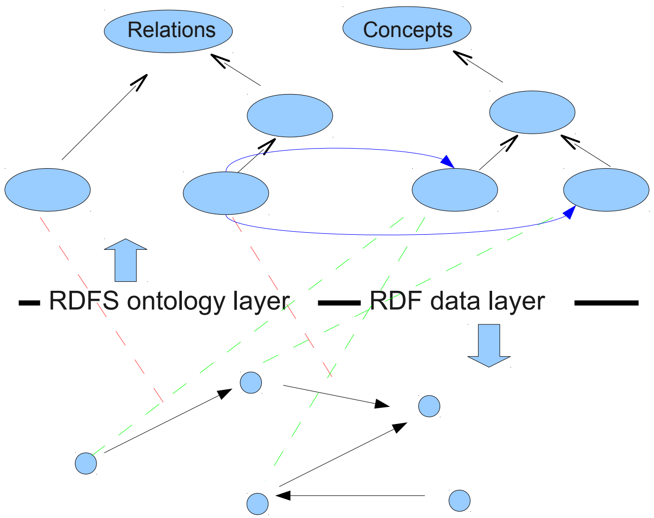
Outline

1. Reminder: RDFS
2. Description Logics
3. Introduction to OWL

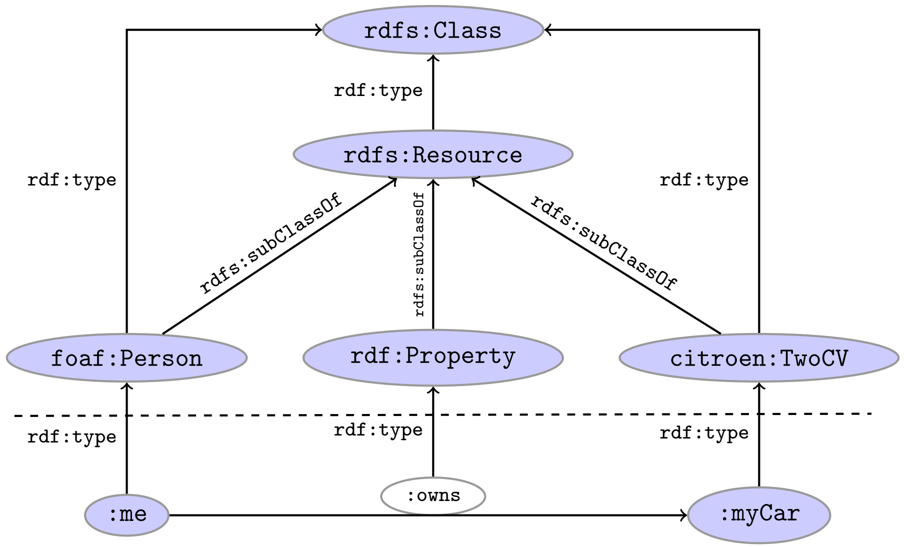
The RDFS vocabulary

- 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.
- There are rules to reason about classes

An RDFS knowledge base



Example



It's complicated

- No clear ontology/data boundary
  - Can have relations between classes and relations

```
:myCar rdf:type citroen:TwoCV.
citroen:TwoCV rdf:type cars:ModelClass.
```

  - Remember: in RDF, properties are resources
  - So they can be subject or object of triples
  - Well, in RDFS, classes are resources
  - So they can also be subject or object of triples
- Incomplete reasoning
  - E.g. can't derive all subtype statements that are semantically valid
  - rdfs:Class not quite the same as a set

Outline

Make it simple!

- “Data level” with resources
- “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 data objects
- 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*

Example: The $\mathcal{ALC}$ Description Logic

Vocabulary

Fix a set of *atomic concepts* A and of *roles* R

$\mathcal{ALC}$ concept descriptions

$$
\begin{array}{rlll}
C, D \rightarrow & A & | & \text{(atomic concept)} \\
& \top & | & \text{(universal concept)} \\
& \bot & | & \text{(bottom concept)} \\
& \neg C & | & \text{(atomic negation)} \\
& C \sqcap D & | & \text{(intersection)} \\
& C \sqcup D & | & \text{(union)} \\
& \forall R.C & | & \text{(value restriction)} \\
& \exists R.C & | & \text{(existential restriction)}
\end{array}
$$

Axioms

$C \sqsubseteq D$ and $C \equiv D$ for concept descriptions D and C.

$\mathcal{ALC}$ Examples

- $\mathit{TwoCV} \sqsubseteq \mathit{Car}$
  - Any 2CV is a car
- $\mathit{TwoCV} \sqsubseteq \forall \mathit{driveAxle}.\mathit{FrontAxle}$
  - All drive axles of 2CVs are front axles
- $\mathit{FrontDrivenCar} \equiv \mathit{Car} \sqcap \forall \mathit{driveAxle}.\mathit{FrontAxle}$
  - A front driven car is one where all drive axles are front axles
- $\mathit{FrontAxle} \sqcap \mathit{RearAxle} \sqsubseteq \bot$ (disjointness)
  - Nothing is both a front axle and a rear axle
- $\mathit{FourWheelDrive} \equiv \exists \mathit{driveAxle}.\mathit{FrontAxle} \sqcap \exists \mathit{driveAxle}.\mathit{RearAxle}$
  - A 4WD has at least one front drive axle and one rear drive axle

$\mathcal{ALC}$ Semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, $A^{\mathcal{I}} \subseteq \Delta$ for each atomic concept A, and $R^{\mathcal{I}} \subseteq \Delta \times \Delta$ for each role R

Interpretation of concept descriptions

$$
\begin{array}{rcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}
\end{array}
$$

Interpretation of Axioms

$C \sqsubseteq D$ holds in $\mathcal{I}$ ($\mathcal{I} \models C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I} \models C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$

Example: Semantics

- Pick a domain $\Delta^\mathcal{I}$ containing all cars, axles of cars, etc.
- Interpret $\mathit{Car}^\mathcal{I} \subseteq \Delta$ as the set of all cars.
- Interpret $\mathit{TwoCV}^\mathcal{I} \subseteq \Delta$ as the set of all 2CV cars.
- Since all 2CV are cars, $\mathit{TwoCV}^\mathcal{I} \subseteq \mathit{Car}^\mathcal{I}$
- Therefore, $\mathit{TwoCV} \sqsubseteq \mathit{Car}$ in this interpretation

But. . .

- Pick a domain $\Delta^\mathcal{J}$ containing fruit and vegetables
- Interpret $\mathit{Car}^\mathcal{J} \subseteq \Delta$ as the set of all fruit.
- Interpret $\mathit{TwoCV}^\mathcal{J} \subseteq \Delta$ as the set of all potatoes.
- Since potatoes are not fruit, $\mathit{TwoCV}^\mathcal{J} \not\subseteq \mathit{Car}^\mathcal{J}$
- Therefore, $\mathit{TwoCV} \sqsubseteq \mathit{Car}$ *does'nt* hold in this interpretation

Existential restrictions

- Let $\Delta^\mathcal{I}$ be the car domain again
- Let $\mathit{driveAxle}^\mathcal{I}$ connect every car with all its drive axles.
- Let $\mathit{FrontAxle}^\mathcal{I}$ be the set of all front axles.
- The interpretation of the concept description

$$\exists \mathit{driveAxle}.\mathit{FrontAxle}$$

is

$$\begin{aligned} & (\exists \mathit{driveAxle}.\mathit{FrontAxle})^\mathcal{I} \\ = \; & \{a \in \Delta^\mathcal{I} \mid \exists b.(a,b) \in \mathit{driveAxle}^\mathcal{I} \wedge b \in \mathit{FrontAxle}^\mathcal{I}\} \end{aligned}$$

- i.e. the set of all things *a* that have a drive axle *b* which is a front axle.

Universal restrictions

- Let $\Delta^\mathcal{I}$ be the car domain again
- Interpret $\mathit{driveAxle}^\mathcal{I}$ and $\mathit{FrontAxle}^\mathcal{I}$ as before
- The interpretation of the concept description

$$\forall \mathit{driveAxle}.\mathit{FrontAxle}$$

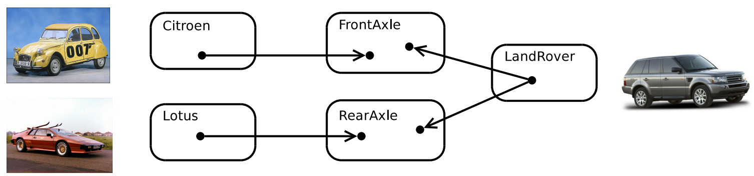
is

$$\begin{aligned} & (\forall \mathit{driveAxle}.\mathit{FrontAxle})^\mathcal{I} \\ = \; & \{a \in \Delta^\mathcal{I} \mid \forall b.(a,b) \in \mathit{driveAxle}^\mathcal{I} \rightarrow b \in \mathit{FrontAxle}^\mathcal{I}\} \end{aligned}$$

- i.e. the set of all things *a* such that all its drive axles *b* are front axles.

Universal and Existential Restrictions cont.

- 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



- In such a model:
  - $\mathit{Citroen} \sqsubseteq \forall \mathit{driveAxle}.\mathit{FrontAxle}$
  - $\mathit{LandRover} \sqsubseteq \exists \mathit{driveAxle}.\mathit{FrontAxle} \sqcap \exists \mathit{driveAxle}.\mathit{RearAxle}$
  - $\mathit{Lotus} \sqsubseteq \forall \mathit{driveAxle}.\mathit{RearAxle}$

Universal Restrictions and rdfs:range

- If the *range* of a role R is C...
- then anything one can reach by R is in C, or...
- for any a and b, if $(a, b) \in R^{\mathcal{I}}$, then $b \in C^{\mathcal{I}}$, or...
- any a is in the interpretation of $\forall R.C$, or
- The axiom $\top \sqsubseteq \forall R.C$ holds
- Ranges can be expressed with universal restrictions
- Example:
  - a drive axle is either a front or a rear axle
  - the range of *driveAxle* is $FrontAxle \sqcup RearAxle$
  - Axiom: $\top \sqsubseteq \forall driveAxle.(FrontAxle \sqcup RearAxle)$

Existential Restrictions and rdfs:domain

- If the *domain* of a role R is C...
- then anything from which one can go by R is in C, or...
- for any a, if there is a b with $(a, b) \in R^{\mathcal{I}}$, then $a \in C^{\mathcal{I}}$, or...
- any a in the interpretation of $\exists R.\top$ is in the interpretation of C, or
- The axiom $\exists R.\top \sqsubseteq C$ holds
- Domains can be expressed with existential restrictions
- Example:
  - a drive axle is something cars have
  - the range of *driveAxle* is *Car*
  - Axiom: $\exists driveAxle.\top \sqsubseteq Car$

Little Boxes

- Historically, description logic axioms and assertions are put in *boxes*
- The TBox
  - is for *terminological knowledge*
  - is independent of any actual instance data
  - for $\mathcal{ALC}$, it is a set of $\sqsubseteq$ axioms
- The ABox
  - is for *assertional knowledge*
  - contains facts about concrete instances $a, b, c, \ldots$
  - A set of concept membership assertions $C(a)$...
  - and role assertions $R(b, c)$

Example TBox and ABox

TBox

$TwoCV \sqsubseteq Car$
$Car \sqsubseteq \exists driveAxle.\top$
$TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
$FrontDrivenCar \equiv Car \sqcap \forall driveAxle.FrontAxle$
$FrontAxle \sqcap RearAxle \sqsubseteq \bot$
$FourWheelDrive \equiv \exists driveAxle.FrontAxle \sqcap \exists driveAxle.RearAxle$

ABox

$TwoCV(myCar)$
$owns(me, myCar)$
$driveAxle(myCar, ax)$
$(FrontAxle \sqcup RearAxle)(ax)$

TBox Reasoning

Model

An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$, written $\mathcal{I} \models \mathcal{T}$, if it satisfies all axioms in $\mathcal{T}$.

- Many reasoning tasks use only the TBox:
- Concept satisfiability: Given C, is there an interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{T}$ and $C^{\mathcal{I}} \neq \emptyset$?
- Concept subsumption: Given C and D, does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold for *every* interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{T}$?
- Concept equivalence: Given C and D, does $C^{\mathcal{I}} = D^{\mathcal{I}}$ hold for *every* interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{T}$?
- Concept disjointness: Given C and D, does $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ hold for *every* interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{T}$?

ABox reasoning

Model

An interpretation $\mathcal{I}$ is a *model* of a TBox and ABox $(\mathcal{T}, \mathcal{A})$, written $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$, if it satisfies all axioms in $\mathcal{T}$ and $\mathcal{A}$.

- ABox consistency:Is there a model of $(\mathcal{T}, \mathcal{A})$?
- Concept membership: Given C and a, does $a^{\mathcal{I}} \in C^{\mathcal{I}}$ hold for *every* interpretation $\mathcal{I}$ with $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$?
- Retrieval: Given C, find all a such that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for *every* interpretation $\mathcal{I}$ with $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$?
- Conjunctive Query Answering (SPARQL)

More Expressive Description Logics

- There are description logics including
  - Axioms about roles (hierarchy, transitivity, etc.)
  - counting role fillers (a car has at least three wheels, etc.)
  - data types (numbers, strings, etc., like literals)
  - etc.
- Won't go into details
- Will see some of these as part of OWL
- Too much expressivity makes reasoning tasks
  - first very expensive
  - then undecidable
- Much research on how much expressivity can be added preserving complexity/decidability

Outline

Quick facts

OWL:

- Acronym for *The Web Ontology Language*.
- Became a W3C recommendation in 2004.
- The undisputed standard ontology language.
- Superseded by OWL 2;
  - 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.

Glimpse ahead: OWL profiles

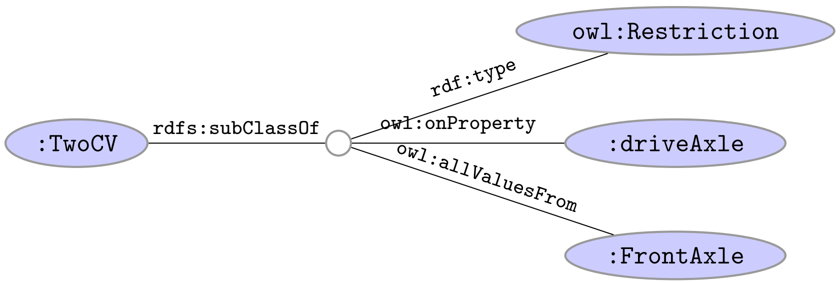
- OWL has various profiles that correspond to different DLs.
- These profiles are tailored for specific ends, e.g.
  - OWL 2 QL:
    - Specifically designed for efficient database integration.
  - OWL 2 EL:
    - A lightweight language with polynomial time reasoning.
    - Much used in medical informatics (e.g. the GALEN ontology).
  - OWL 2 RL:
    - Designed for compatibility with rule-based inference tools.

OWL Syntaxes

- Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- Same for OWL:
- Defined as set of things that can be said about classes, properties, instances
- DL symbols ($\sqcap, \sqcup, \exists, \forall$) hard to find on keyboard
- 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: close to DL, but text, used in some tools

Example: Universal Restrictions in OWL/RDF

- $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$



- In Turtle syntax:

```
:TwoCV rdfs:subClassOf [ rdf:type owl:Restriction ;
                          owl:onProperty :driveAxle ;
                          owl:allValuesFrom :FrontAxle
                        ] .
```

Example: Universal Restrictions in Other Formats

- $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- In OWL/XML syntax:

```
<SubClassOf>
   <Class URI="&cars;TwoCV"/>
   <ObjectAllValuesFrom>
      <ObjectProperty URI="&cars;driveAxle"/>
      <Class URI="&cars;FrontAxle"/>
   </ObjectAllValuesFrom>
</SubClassOf>
```

- In OWL Functional syntax:

```
SubClassOf(CV ObjectAllValuesFrom(driveAxle FrontAxle))
```


Manchester OWL Syntax

- Used in Protégé for concept descriptions
- Also has a syntax for axioms, less used
- Correspondence to DL constructs:

DL	Manchester
$C \sqcap D$	C and D
$C \sqcup D$	C or D
$\neg C$	not C
$\forall R.C$	R only C
$\exists R.C$	R some C

- Examples:

DL	Manchester
$FrontAxle \sqcup RearAxle$	FrontAxle or RearAxle
$\forall driveAxle.FrontAxle$	driveAxle only FrontAxle
$\exists driveAxle.RearAxle$	driveAxle some RearAxle

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 2CV
- Classify.
- Show that Sahara is equivalent to bottom.
- Explain why. In particular, disjointness of front and rear axles

The Relationship to Description Logics

- Protégé presents ontologies almost like an OO modelling tool
- Everything can be mapped to DL axioms!
- (will see some features that require more than $\mathcal{ALC}$ next time)
- We have seen how domain and range become ex./univ. restrictions
- C and D disjoint: $C \sqsubseteq \neg D$
- Many ways of saying the same thing in OWL, more in Protégé
- Reasoning (e.g. Classification) maps everything to DL first

OWL in Jena

- Can use usual Jena API to build OWL/RDF ontologies
- Cumbersome and error prone!
- Jena class `OntModel` provides convenience methods to create OWL/RDF ontologies.
- e.g.

```
OntModel model = ModelFactory.createOntologyModel();
Property driveAxle = model.createProperty(CARS+"driveAxle");
OntClass car = model.createClass(CARS+"Car");
OntClass frontAxle = model.createClass(CARS+"FrontAxle");
Resource r = model.createAllValuesFromRestriction(
                      null, driveAxle, frontAxle);
car.addSuperClass(r);
```

- Can be combined with inferencing mechanisms from previous lecture
  - See class OntModelSpec

The OWL API

- OWL in Jena means OWL expressed as RDF
- Still somewhat cumbersome, tied to OWL/RDF peculiarities
- For pure ontology programming, consider OWL API:

  `http://owlapi.sourceforge.net/`

- Works on the level of concept descriptions and axioms
- Can parse and write all mentioned OWL formats, and then some

Next time

- More about OWL. . .
- Saying that things are the same or not
- More about roles/properties:
  - object properties and datatype properties
  - transitive, inverse, symmetric, functional properties