

# INF3580 – Semantic Technologies – Spring 2010

## Lecture 9: More OWL, Role modeling

Audun Stolpe

23rd March 2010



DEPARTMENT OF  
INFORMATICS



UNIVERSITY OF  
OSLO

## Generelle opplysninger:

### Om obligene:

- oblig 1 er rettet
- e-post skal være sendt ut til alle som har levert
- frist for ny levering 8. april
- kommentarer ligger ute på kursets hjemmeside
- sammen med enkelte hint til løsningen

### Angående sommeren:

- Bli betalt for å jobbe med semantisk teknologi!
- Vi trenger studenter til å lære seg *Cambridge Semantics*
- Ta kontakt med Martin eller Audun.

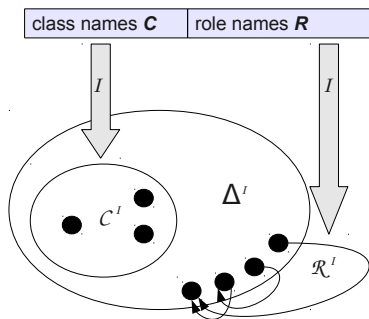
## Today's Plan

- 1 Reminder: OWL
- 2 Role modeling
- 3 A worked example

## Outline

- 1 Reminder: OWL
- 2 Role modeling
- 3 A worked example

## Schematic representation of OWL/DL interpretations



- No reference/extension distinction
- That is, no function  $IEXT$
- No properties in the domain
- Classes are sets
- Properties are relations
- Simple extensional semantics

 $\mathcal{ALC}$  Semantics

## Interpretation

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

## Interpretation of concept descriptions

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

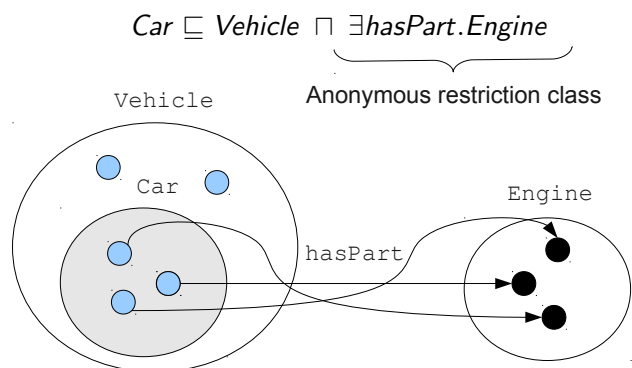
 $\mathcal{ALC}$  TBox and ABox

- The TBox
  - is for *terminological knowledge*
  - is independent of any actual instance data
  - is a set of  $\sqsubseteq$  axioms
- The ABox
  - is for *assertional knowledge*
  - contains facts about concrete instances  $a, b, c, \dots$
  - A set of concept assertions  $C(a) \dots$
  - and role assertions  $R(b, c)$

## Recap of restrictions

- Existential restrictions
  - have the form  $\exists R.C$
  - typically used to connect classes
  - $A \sqsubseteq \exists R.C$ : Every  $A$ -object is  $R$ -related to *some*  $C$ -object
- Universal restrictions
  - have the form  $\forall R.C$
  - restrict the things a type of object can be connected to
  - $A \sqsubseteq \forall R.C$ : Every  $A$ -object is  $R$ -related to  $C$ -objects *only*
  - $A$ -objects may not be  $R$ -related to anything at all
- Example:
  - A car is a motorised vehicle
  - $Car \sqsubseteq Vehicle \sqcap \exists hasPart.Engine$

## Existential restrictions illustrated



## A different perspective

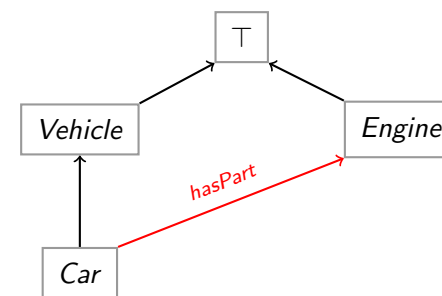


Figure: Connecting classes

## Outline

- 1 Reminder: OWL
- 2 Role modeling
- 3 A worked example

## Role characteristics and relationships

Role characteristics are mathematical properties of roles.

- A role can be:
  - reflexive/irreflexive
  - symmetric/asymmetric
  - transitive
  - functional/inverse functional

Role relationships: Roles  $R$  and  $S$  can be

- declared *disjoint*, meaning that  $R^I \cap S^I = \emptyset$
- related as *inverses*, meaning that  $S^I = (R^-)^I$
- subsumed under each other, meaning that  $R^I \subseteq S^I$
- chained, e.g.  $R^I \circ S^I \subseteq S^I$

## Corresponding mathematical properties and operations

A relation  $R$  over a set  $X$  is

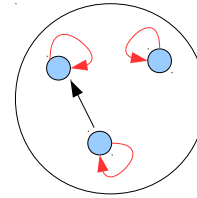
- Reflexive:** if  $(a, a) \in R$  for all  $a \in X$   
**Irreflexive:** if  $a \in X$  implies  $(a, a) \notin R$   
**Symmetric:** if  $(a, b) \in R$  implies  $(b, a) \in R$   
**Asymmetric:** if  $(a, b) \in R$  implies  $(b, a) \notin R$   
**Transitive:** if  $(a, b), (b, c) \in R$  implies  $(a, c) \in R$   
**Functional:** if  $(a, b), (a, c) \in R$  implies  $b = c$   
**Inverse functional:** if  $(a, b), (c, b) \in R$  implies  $a = c$

If  $R$  and  $S$  are binary relations on  $X$  then

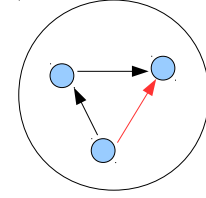
- $(a, c) \in R \circ S$ : if  $(a, b) \in R$  and  $(b, c) \in S$  for some  $b \in X$   
 $(b, a) \in R^-$ : if  $(a, b) \in R$ .

## Relation diagrams

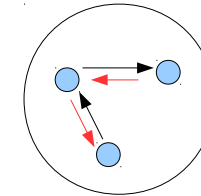
A reflexive relation:



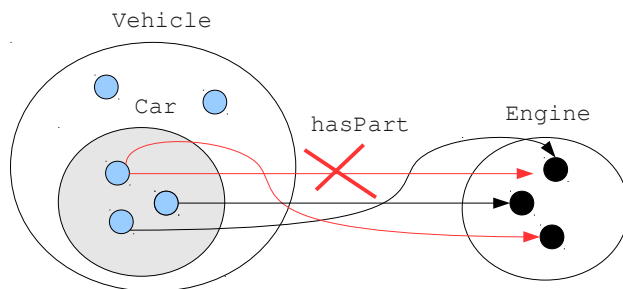
A transitive relation:



A symmetric relation:

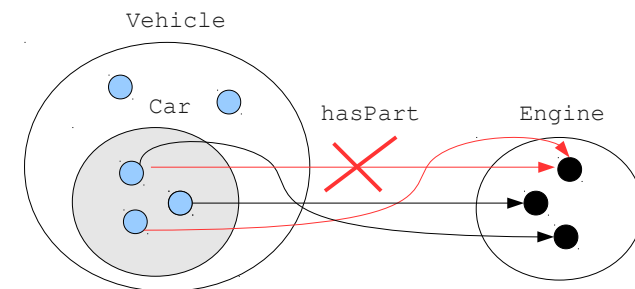


## Functionality



A (normal) car doesn't have more than one engine

## Inverse functionality



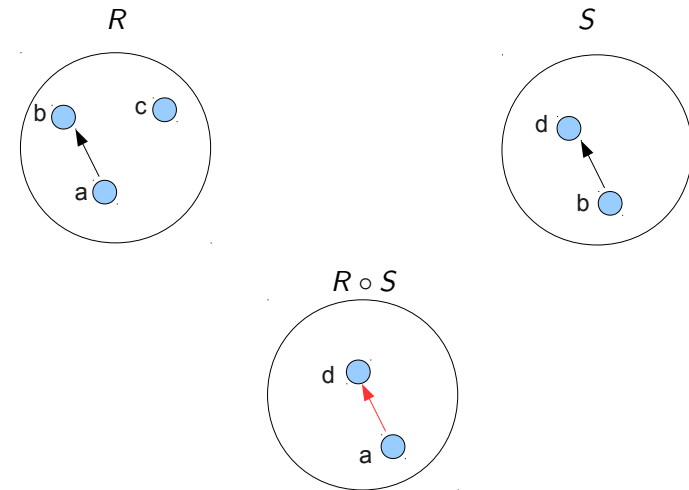
An engine doesn't sit in more than one car (simultaneously)

## Some role relationships: Inverses



Inverse roles  $R$  and  $R^{-}$ .

## Chaining of roles



## Some relations from ordinary language

- Symmetric relations:
  - \_ sibling of \_
  - \_ different from \_
- Non-symmetric relations:
  - \_ brother of \_
  - \_ likes \_
- Asymmetric relations:
  - \_ taller than \_ (under a strict interpretation)
  - \_ member of \_
- Transitive relations:
  - \_ taller than \_
  - \_ part of \_ (under certain qualifications)
- Functional relations:
  - \_ was born by \_
- Inverse functional relations:
  - \_ gave birth to \_

## Som inverses and chains

Some inverses:

- Uncle/nephew
- Gave birth to/was born by
- To the left of/to the right of
- Taller than/shorter than
- etc.

Some role chains:

- $\text{fatherOf} \circ \text{brotherOf} \sqsubseteq \text{uncleOf}$
- $\text{isLocatedIn} \circ \text{isPartOf} \sqsubseteq \text{isLocatedIn}$

## Datatype properties and object properties

OWL enforces a separation between datatype- and object properties:

Object properties:

- Also known as *abstract roles*
- connect objects with objects
- Example in Turtle syntax:  

```
foaf:knows a owl:ObjectProperty .
```

Datatype properties:

- Also known as *concrete roles*
- connect objects with literal values, i.e. with elements of datatypes.
- Example in Turtle-syntax:  

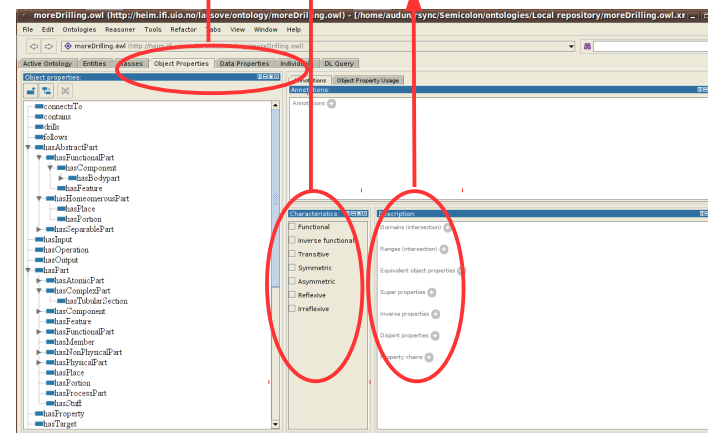
```
ex:age a owl:DatatypeProperty .
ex:age rdfs:range xsd:positiveInteger .
```

## Managing roles in Protege

Object/datatype property tabs

Role characteristics

Domain/range, role relationships

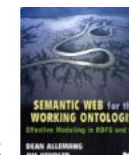


## Outline

- 1 Reminder: OWL
- 2 Role modeling
- 3 A worked example

## Example: Merging product information

The example is an adaptation from Allemang and Hendler:  
 "Semantic Web for the Working Ontologist":



Suppose we want to integrate product information, and that

- data is stored in two different tables
- in two different databases
- one contains information about the product per se
- and the other about the facilities needed to produce them

## Table excerpts I

Product				
ID	Model Number	Division	Manufacture Location	Available
1	ZX-3	Manufacturing	Sacramento	23
2	ZX-3P	Manufacturing	Seoul	14
3	ZX-3S	Support	Hong Kong	100
4	B1431	Control	Hong Kong	4
5	B1430X	Engineering	Elizabeth	14
6	DBB-12	Accessories	Cleveland	87

Figure: Table of products

## Table excerpts II

Product		
ID	Model Number	Facility
1	B1430X	Assembly Center
2	1180-M	Machine Shop
3	TC-43	Factory
4	ZX-3P	Factory
5	B1431	Assembly Center
6	SP-1234	Machine Shop

Figure: Parts and the facilities required to produce them

## The challenge

We wish to integrate the two tables, so that e.g.

- places can be correlated with production facilities

However, we would like to do so in manner such that

- we do not have to go through the rows one-by-one
- in a manual editing process

Rather we would like to

- Specify a set of general relationships between the respective columns
- that enables a reasoner to *infer* the correlations whenever they exist



## Exposing RDBs as RDF

Information in a table can be encoded as RDF:

The recipe is:

- 1 Come up with a URI for the database as such, and in this namespace:
  - Make each row in the table a resource,
  - construct the resource name from the table name and the primary key
- 2 make each cell a triple where
  - the resource corresponding to the row is the subject of the triple
  - the predicate name is constructed from the table and column name
  - the cell value is the object of the triple

This is called *exposing RDBs as RDF* and can be done by several tools:

For instance:

- D2RQ
- SquirrelRDF
- OpenLink Virtuoso

## Desirable features

These tools have one or more of the following features

- the data is exposed as *virtual RDF*,
- that is, conversion is on-demand rather than up-front
- they offer general-purpose mapping from RDB to ontology
- that is, tables can be mapped to classes of one's own choosing
- and columns can be mapped to properties

D2RQ, for one, has all features.



## The RDF encoding

There are  $5 \times 6 = 30$  triples for the first table, among others

### Manufacture location triples

```
mf:Product1 mf:Product_Manufacture_location "Sacramento" .
mf:Product2 mf:Product_Manufacture_location "Seoul" .
mf:Product3 mf:Product_Manufacture_location "Hong Kong" .
mf:Product4 mf:Product_Manufacture_location "Elizabeth" .
mf:Product5 mf:Product_Manufacture_location "Hong Kong" .
mf:Product6 mf:Product_Manufacture_location "Cleveland" .
```

We assume that `mf:` abbreviates the namespace of the database.

## .. contd

Similarly there are  $3 \times 6 = 18$  triples for the second table, among others

### Production facility triples

```
p:Product1 p:Product_Facility "Assembly Center" .
p:Product2 p:Product_Facility "Machine Shop" .
p:Product3 p:Product_Facility "Factory" .
p:Product4 p:Product_Facility "Factory" .
p:Product5 p:Product_Facility "Assembly Center" .
p:Product6 p:Product_Facility "Machine Shop" .
```

We assume that `p:` abbreviates the namespace of the database.

## Solution

The challenge can now be solved by a two-step procedure:

1. Declare the respective **Model Number** columns equivalent properties:
  - if a product  $x$  has a `mf:Model_Number` value of "ZX-3P"
  - then  $x$  also has the same value for `p:Model_Number`
- This can be done manually, by adding the following triples:
 

```
mf:Product_Number rdfs:subPropertyOf p:Product_Number .
p:Product_Number rdfs:subPropertyOf mf:Product_Number .
```
- or it can be done in Protegé



## solution contd.

2. Declare one property to be *inverse functional*

- The range of such a property can be considered a set of unique keys
- i.e. elements of the range provide unique identifiers for each element of the domain.

Thus,

- If, say, `mf:Model_Number` is declared to be inverse functional,
- then records with the same `mf:Model_Number` represent the same product,

Inverse functionality,

- can be declared manually by adding a triple such as

```
mf:Model_Number a owl:InverseFunctionalProperty .
```

which will land you in OWL-full

- or by using the `owl:hasKey` facility of OWL 2 (consult the spec)

## A sample trace

## A SPARQL query

```
SELECT ?location ?facility WHERE{
    ?product mf:Manufacture_Location ?location .
    ?product p:Product_Facility ?facility.
}
```

- SPARQL finds `mf:Product4`
- which has `mf:Manufacture_Location` "Hong Kong"
- and `mf:Product_Number` "B1431"

## trace contd.

- "B1431" is also the `p:Product_Number` of `p:Product5`
- these properties are equivalent
- so "B1431" is also the `mf:Product_Number` of `p:Product5`
- whence, since `mf:Product_Number` is inverse functional, we have `p:Product5 = mf:Product4`
- now, `p:Product5` has `p:Product_Facility` "Assembly Center",
- and `mf:Product4` has `mf:Manufacture_Location` "Hong Kong"
- So ("Hong Kong", "Assembly Center") is a solution for the query

## Other common role modeling patterns

- Transitivity and reflexivity for ordering relations, e.g.
  - the mereological notion of part-whole
  - being a part of a part of is being a part of
  - everything is part of itself
- Inversely related ordering relations, e.g.
  - `hasPart` and `partOf`
  - if a has b as a part then b is a part of a
- Asymmetry for strict ordering relations, e.g.
  - the mereological `isProperPartOf`
  - if a is a proper part of b then b cannot be a proper part of a
- Functional properties where sameness should be inferred, e.g.
  - the `hasFather` relation,
  - where fathers may be known by different names