

Semantic Days 2010 Tutorial

Semantic Web Technologies

Lecture 4: OWL, the Web Ontology Language

Martin Giese

31st May 2010



DEPARTMENT OF
INFORMATICS



UNIVERSITY OF
OSLO

Outline

1 The RDFS vocabulary

2 OWL

Modeling

- Remember what we said about modeling in the intro. . .

Modeling

- Remember what we said about modeling in the intro...
 - Fix a vocabulary

Modeling

- Remember what we said about modeling in the intro. . .
 - Fix a vocabulary
 - Describe interaction of terms in the vocabulary

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

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

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

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

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

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

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

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

The RDFS vocabulary

- RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- A predefined vocabulary allows statements about classes

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:

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.

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.

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`)

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:

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

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.

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.

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.

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*

Semantics

- The meaning of this vocabulary is given through *inference rules*
- Given some statements, a rule allows to infer other statements

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

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:

$$\frac{C \text{ rdfs:subClassOf } D \quad x \text{ rdf:type } C}{x \text{ rdf:type } D}$$

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:

$$\frac{C \text{ rdfs:subClassOf } D \quad x \text{ rdf:type } C}{x \text{ rdf:type } D}$$

- Example of inference:

$$\frac{\text{:City rdfs:subClassOf :Place} \quad \text{:Oslo rdf:type :City}}{\text{:Oslo rdf:type :Place}}$$

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:

$$\frac{C \text{ rdfs:subClassOf } D \quad x \text{ rdf:type } C}{x \text{ rdf:type } D}$$

- Example of inference:

$$\frac{\text{:City rdfs:subClassOf :Place} \quad \text{:Oslo rdf:type :City}}{\text{:Oslo rdf:type :Place}}$$

- Asserted and inferred triples

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:

$$\frac{C \text{ rdfs:subClassOf } D \quad x \text{ rdf:type } C}{x \text{ rdf:type } D}$$

- Example of inference:

$$\frac{\text{:City rdfs:subClassOf :Place} \quad \text{:Oslo rdf:type :City}}{\text{:Oslo rdf:type :Place}}$$

- **Asserted** and inferred triples

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:

$$\frac{C \text{ rdfs:subClassOf } D \quad x \text{ rdf:type } C}{x \text{ rdf:type } D}$$

- Example of inference:

$$\frac{\text{:City rdfs:subClassOf :Place} \quad \text{:Oslo rdf:type :City}}{\text{:Oslo rdf:type :Place}}$$

- Asserted and **inferred** triples

Rules for Properties

- Sub-properties:

$$\frac{R \text{ rdfs:subPropertyOf } S \quad x R y}{x S y}$$

Rules for Properties

- Sub-properties:

```
:marriedTo rdfs:subPropertyOf foaf:knows  
:martin :marriedTo :simone  
-----  
:martin foaf:knows :simone
```

- Domain:

Rules for Properties

- Sub-properties:

$$\frac{\begin{array}{l} \text{:marriedTo rdfs:subPropertyOf foaf:knows} \\ \text{:martin :marriedTo :simone} \end{array}}{\text{:martin foaf:knows :simone}}$$

- Domain:

$$\frac{\begin{array}{l} R \text{ rdfs:domain } C \\ x R y \end{array}}{x \text{ rdf:type } C}$$

- Range:

Rules for Properties

- Sub-properties:

```

:marriedTo rdfs:subPropertyOf foaf:knows
           :martin :marriedTo :simone
-----
           :martin foaf:knows :simone

```

- Domain:

```

foaf:knows rdfs:domain foaf:Person
           :martin foaf:knows :simone
-----
           :martin rdf:type foaf:Person

```

- Range:

Rules for Properties

- Sub-properties:

$$\frac{\begin{array}{l} \text{:marriedTo rdfs:subPropertyOf foaf:knows} \\ \text{:martin :marriedTo :simone} \end{array}}{\text{:martin foaf:knows :simone}}$$

- Domain:

$$\frac{\begin{array}{l} \text{foaf:knows rdfs:domain foaf:Person} \\ \text{:martin foaf:knows :simone} \end{array}}{\text{:martin rdf:type foaf:Person}}$$

- Range:

$$\frac{\begin{array}{l} R \text{ rdfs:range } C \\ x R y \end{array}}{y \text{ rdf:type } C}$$

Rules for Properties

- Sub-properties:

```

:marriedTo rdfs:subPropertyOf foaf:knows
           :martin :marriedTo :simone
-----
           :martin foaf:knows :simone

```

- Domain:

```

foaf:knows rdfs:domain foaf:Person
           :martin foaf:knows :simone
-----
           :martin rdf:type foaf:Person

```

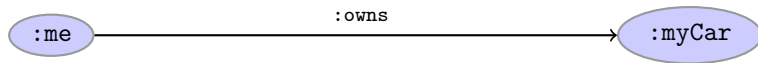
- Range:

```

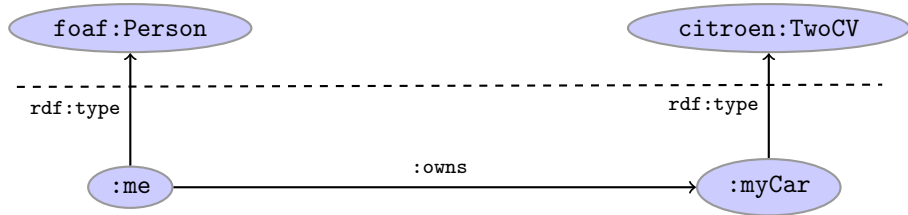
foaf:knows rdfs:range foaf:Person
           :martin foaf:knows :simone
-----
           :simone rdf:type foaf:Person

```

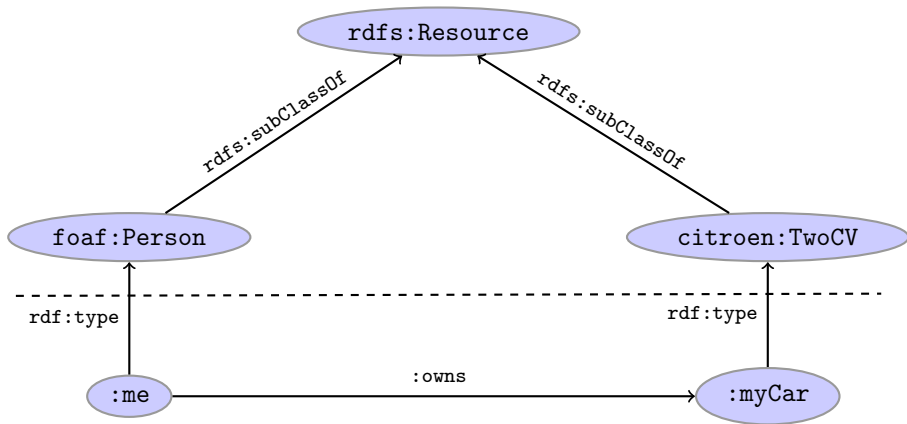
Example



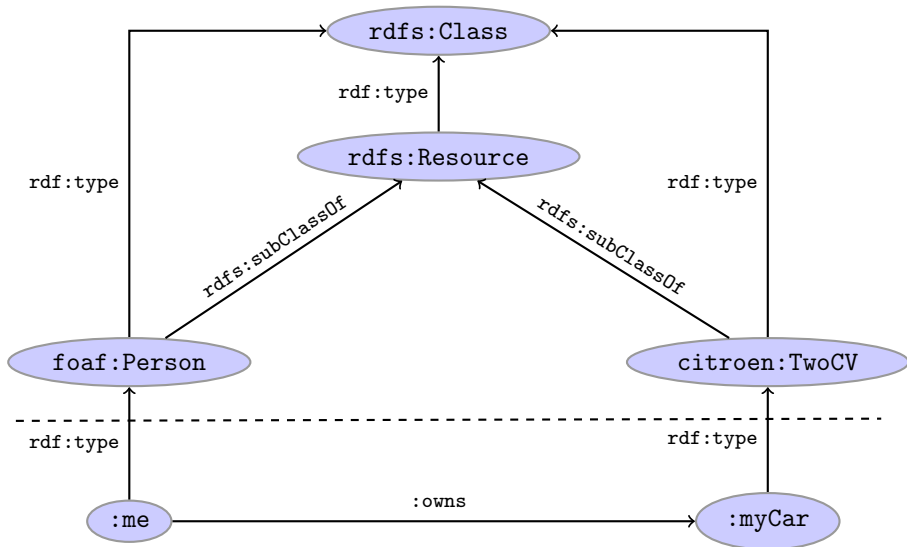
Example



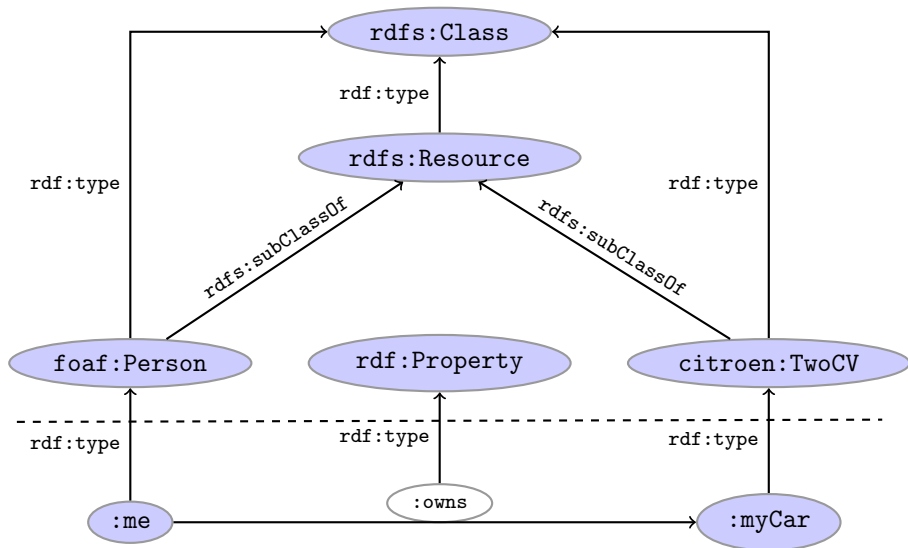
Example



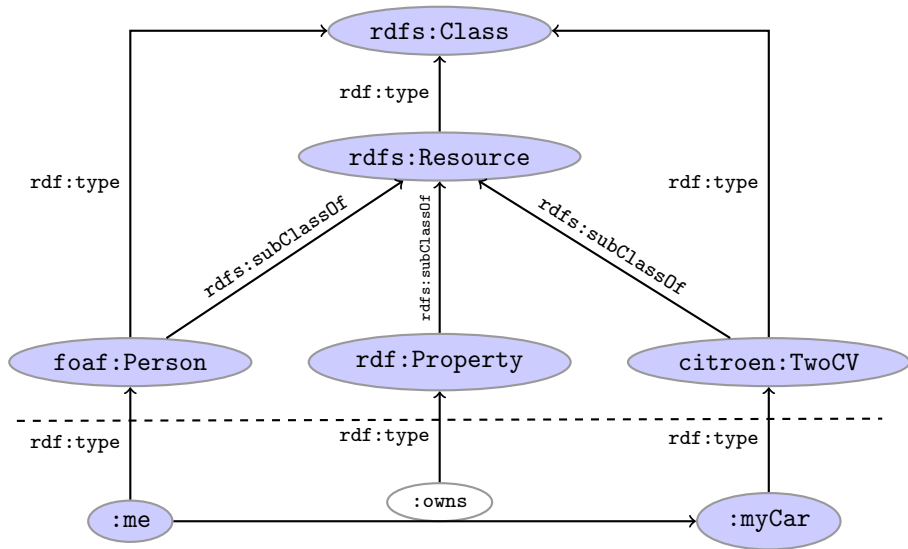
Example



Example



Example



Outline

1 The RDFS vocabulary

2 **OWL**

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “Data level” with resources

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “Data level” with resources
- “Ontology level” with properties and “classes”

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “Data level” with resources
- “Ontology level” with properties and “classes”
- Classes and properties *not* part of the domain!

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “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

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “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

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “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

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “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:

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “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

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “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

OWL is simpler. . .

- OWL is an extension of a restriction of RDFS
- No types of types of types in OWL!
- “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*

OWL Quick Facts

OWL:

- Acronym for *The Web Ontology Language*.



OWL Quick Facts

OWL:

- Acronym for *The Web Ontology Language*.
- Became a W3C recommendation in 2004.



OWL Quick Facts

OWL:

- Acronym for *The Web Ontology Language*.
- Became a W3C recommendation in 2004.
- The undisputed standard ontology language.



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;



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;
 - a backwards compatible extension that adds new capabilities.



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;
 - a backwards compatible extension that adds new capabilities.
- OWL is a language to express “ontologies”



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;
 - a backwards compatible extension that adds new capabilities.
- OWL is a language to express “ontologies”
- i.e. express facts about a domain, like RDFS



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;
 - 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



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;
 - 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.)



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;
 - 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,...

OWL Syntaxes

- Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- Same for OWL:

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 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

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 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

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

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

OWL Concept Descriptions

- In OWL, classes can be combined to form new concepts
- Boolean combinations:

OWL Concept Descriptions

- In OWL, classes can be combined to form new concepts
- Boolean combinations:
 - (foaf:Person and not gender:Female) or rodents:Mouse

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

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:

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

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

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:

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

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

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.

OWL Axioms

- Concept descriptions don't say anything.
- There are essentially two kinds of things you can say in OWL:

OWL Axioms

- Concept descriptions don't say anything.
- There are essentially two kinds of things you can say in OWL:
- Concept membership

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`)

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)`.

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

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`)

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)`

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

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

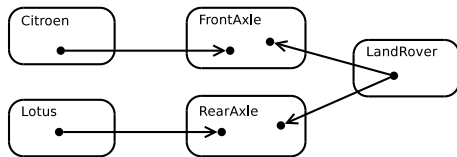
OWL Axioms

- 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:

Object Properties vs. Datatype Properties

- Can use RDF properties with resource or literal objects:
 - `:martin foaf:knows :simone .`

Object Properties vs. Datatype Properties

- Can use RDF properties with resource or literal objects:
 - `:martin foaf:knows :simone .`
 - `:martin foaf:knows "Simone" .`

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.

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 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

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

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

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

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

Exercise: OWL, Protégé

- Create ontology using Protégé.
- First exercise basic step by step editing.
- Second exercise more advanced, using constructs not showed in this lecture.
- Reasoning.

Go to <http://sws.ifi.uio.no/semdays2010/> for the full exercise text.