

OWL
Ontology Web Language
(part 1)

Maciej Janik

Semantic Web
2009-07-02



- OWL
 - ◆ Ontology Web Language

 - ◆ W3C Recommendation 10 Feb 2004
 - ◆ it is built on top of RDF and RDFS
 - ◆ have much stronger expressive power
 - ◆ defines formal semantics
 - ◆ supports reasoning (with use of different flavors of logic)

 - ◆ **was designed** to be interpreted by computers
 - ◆ **was not designed** for being read by people

When RDFS is not enough

- Global and local scope of properties
 - ◆ **rdfs:range** defines the range of a property for all classes that use this property
 - e.g. animals eat plants and meat, but goat eats only plants
 - ◆ It is not possible to redefine property restrictions for specific classes in RDFS
 - we have to define new property, with new range restrictions
 - where to put it in property hierarchy? does it make sense?

- Specific logic associated with properties
 - ◆ transitivity (e.g. “less than”)
 - ◆ uniqueness (e.g. “is president of”)
 - ◆ inversion of another relationship (e.g. “teaches” and “is taught by”)

- Cardinality restrictions
 - ♦ course is taught by **at least** one professor
 - ♦ bridge card game can be played by **exactly** 4 players
- Disjoint classes
 - ♦ Gender can be either **male** or **female**, never both
- Boolean combinations of classes (like in set theory)
 - ♦ Create new classes by combining other classes using **union**, **intersection**, and **complement**
 - person** = union of disjoint classes **male** and **female**
 - vegetarian food** = complement of **meat food**

We need more expressiveness and reasoning

Expressive Power Vs Reasoning

- We would like to have maximum expressive power and support for sophisticated reasoning
... but ...
- *The richer the language is, the more inefficient the reasoning support becomes*
- It may even become *not computable*
- Useful language must compromise:
 - ◆ support by reasonably efficient reasoners
 - ◆ enough expressiveness for large classes of ontologies and knowledge

What is reasoning?

- Class inheritance and membership
- Equivalent classes
 - ◆ A is equivalent to B, and B is equivalent to C, so we know that A is equivalent to C.
- Classification
 - ◆ Certain property-value pairs may be defined as sufficient condition for membership in class A
 - e.g. domain or range definition
 - ◆ if an instance x satisfies given property-value, we infer that x must be an instance of class A
- Consistency checking
 - ◆ X is an instance of **both** classes A and B
 - ◆ A and B are **disjoint** (cannot have common members)
 - This is an indication of an error in the ontology

- Use of reasoning
 - ◆ checking consistency of the ontology
 - ◆ checking taxonomy relationship (detect cycles, disjointness or other restrictions)
 - ◆ checking for unintended relationships between classes
 - ◆ automatically classifying instances to classes
 - ◆ designing complicated and large ontologies, specifically in collaborative environment
 - ◆ integrating multiple ontologies, so they present consistent description of the world

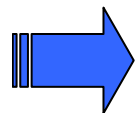
Reasoning Support for OWL

- Specific semantics is required to support reasoning
- Formal semantics and reasoning support are usually provided by
 - ◆ mapping an ontology language to a known logical formalism
 - ◆ using automated reasoners that already exist for those formalisms
- OWL is (partially) mapped on a description logic
 - ◆ use of existing reasoners (e.g. RACER)
- Description logics are a subset of predicate logic for which **efficient reasoning support is possible**

Flavors of OWL

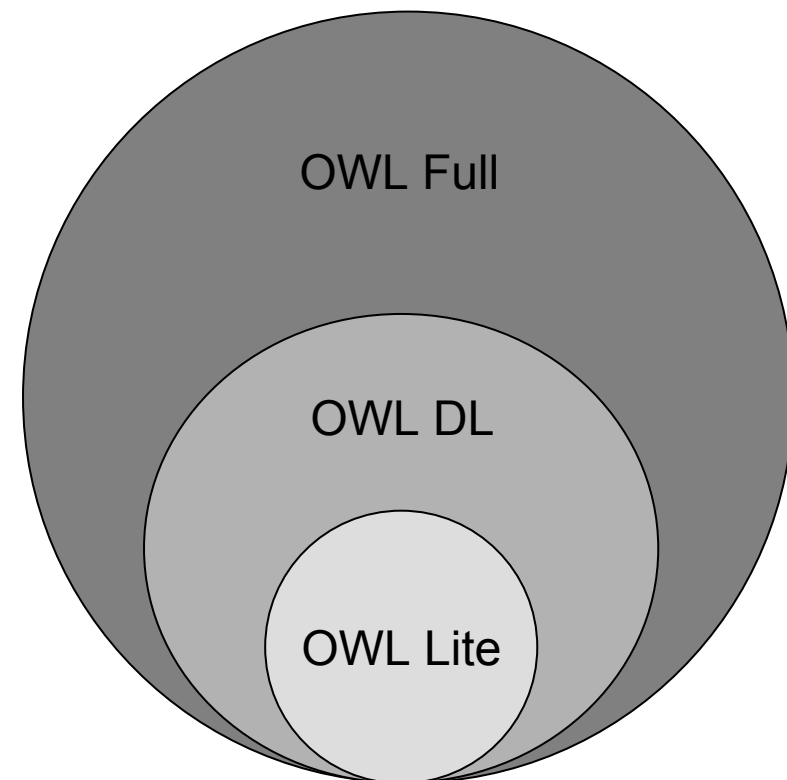
- W3C's Web Ontology Working Group defined OWL as three different sublanguages:

- ◆ OWL Full



- ◆ OWL DL (Description Logic)

- ◆ OWL Lite



Namespace

`xmlns:owl = "http://www.w3.org/2002/07/owl#"`

- Uses **all** of the OWL languages primitives to express facts and reason about them
- Allows combining primitives from RDF and RDFS with OWL in any configuration – combining different kind of semantics
- Fully upward-compatible with RDF (both syntactically and semantically)
- It is so powerful that it is in some cases **undecidable**
 - ◆ As a result, there is no complete (or efficient) reasoning support

- OWL DL – based on Description Logic
- Sublanguage of OWL Full
 - ◆ Restricts application of the constructors from OWL and RDF
 - ◆ Application of OWL's constructors' to each other is disallowed
 - ◆ Limited vocabulary and semantics
- OWL DL **permits efficient reasoning support**
- **But** it comes with a cost of full compatibility with RDF:
 - ◆ Every legal OWL DL document is a legal RDF document.
 - ◆ Not every RDF document is a legal OWL DL document.

- Further restricted expressivity of OWL DL
- Excluding specific language constructions to simplify computability
 - ◆ enumerated classes,
 - ◆ disjointness statements,
 - ◆ arbitrary cardinality (only 0 or 1).
- The gains:
 - ◆ Easier for users to grasp and understand,
 - ◆ Easier to implement and use on the site.

RDF Schema Features:

- [Class \(Thing, Nothing\)](#)
- [rdfs:subClassOf](#)
- [rdf:Property](#)
- [rdfs:subPropertyOf](#)
- [rdfs:domain](#)
- [rdfs:range](#)
- [Individual](#)

(In)Equality:

- [equivalentClass](#)
- [equivalentProperty](#)
- [sameAs](#)
- [differentFrom](#)
- [AllDifferent](#)
- [distinctMembers](#)

Property Characteristics:

- [ObjectProperty](#)
- [DatatypeProperty](#)
- [inverseOf](#)
- [TransitiveProperty](#)
- [SymmetricProperty](#)
- [FunctionalProperty](#)
- [InverseFunctionalProperty](#)

Property Restrictions:

- [Restriction](#)
- [onProperty](#)
- [allValuesFrom](#)
- [someValuesFrom](#)

Restricted Cardinality:

- [minCardinality](#) (only 0 or 1)
- [maxCardinality](#) (only 0 or 1)
- [cardinality](#) (only 0 or 1)

Header Information:

- [Ontology](#)
- [imports](#)

Class Intersection:

- [intersectionOf](#)

Versioning:

- [versionInfo](#)
- [priorVersion](#)
- [backwardCompatibleWith](#)
- [incompatibleWith](#)
- [DeprecatedClass](#)
- [DeprecatedProperty](#)

Annotation Properties:

- [rdfs:label](#)
- [rdfs:comment](#)
- [rdfs:seeAlso](#)
- [rdfs:isDefinedBy](#)
- [AnnotationProperty](#)
- [OntologyProperty](#)

Datatypes

- [xsd datatypes](#)

Class Axioms:

- [oneOf, dataRange](#)
- [disjointWith](#)
- [equivalentClass](#)
(applied to class expressions)
- [rdfs:subClassOf](#)
(applied to class expressions)

Boolean Combinations of Class Expressions:

- [unionOf](#)
- [complementOf](#)
- [intersectionOf](#)

Arbitrary Cardinality:

- [minCardinality](#)
- [maxCardinality](#)
- [cardinality](#)

Filler Information:

- [hasValue](#)

xsd:string	xsd:normalizedString	xsd:boolean	
xsd:decimal	xsd:float	xsd:double	
xsd:integer	xsd:nonNegativeInteger	xsd:positiveInteger	
xsd:nonPositiveInteger	xsd:negativeInteger		
xsd:long	xsd:int	xsd:short	xsd:byte
xsd:unsignedLong	xsd:unsignedInt	xsd:unsignedShort	xsd:unsignedByte
xsd:hexBinary	xsd:base64Binary		
xsd:dateTime	xsd:time	xsd:date	xsd:gYearMonth
xsd:gYear	xsd:gMonthDay	xsd:gDay	xsd:gMonth
xsd:anyURI	xsd:token	xsd:language	
xsd:NMTOKEN	xsd:Name	xsd:NCName	

XML Schema allows construction of user-defined data types (even complex ones)
e.g., the data type of **adultAge** includes all integers greater than 18

OWL **do not allow** such datatypes (only the listed ones)

```
<owl:Ontology rdf:about=„“>
```

```
<rdfs:comment
```

```
  rdf:datatype=„http://www.w3.org/2001/XMLSchema#string“>
```

```
  SWRC Ontology, Version from December 2005</rdfs:comment>
```

```
<owl:versionInfo>v0.5</owl:versionInfo>
```

```
<owl:imports rdf:resource=„http://www.example.org/foo“/>
```

Importing other ontologies from web
Directly reusing other ontologies

```
<owl:priorVersion rdf:resource=„http://ontoware.org/projects/swrc“'/>
```

```
</owl:Ontology>
```

```
<rdf:Description rdf:about=„Professor“>  
  <rdf:type rdf:resource=„&owl;Class“/>  
</rdf:description>
```

OR

```
<owl:Class rdf:about=„Professor“/>
```

Defining hierarchy

```
<owl:Class rdf:about=„Professor“>  
  <rdfs:subClassOf rdf:resource=„Lecturer“/>  
</owl:Class>
```

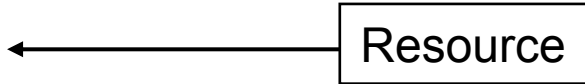
```
<owl:Class rdf:about=„Lecturer“>  
  <rdfs:subClassOf rdf:resource=„Person“/>  
</owl:Class>
```

```
<owl:ObjectProperty rdf:about=„teaches“/>
```

```
<owl:DatatypeProperty rdf:about=„hasPersonalNumber“/>
```

```
<owl:ObjectProperty rdf:about=„teaches“>  
  <rdfs:domain rdf:resource=„Professor“/>  
  <rdfs:range rdf:resource=„Lecture“/>  
</owl:ObjectProperty>
```

Resource



```
<owl:DatatypeProperty rdf:about=„hasPersonalNumber“>  
  <rdfs:domain rdf:resource=„Professor“/>  
  <rdfs:range rdf:resource=„&xsd;positiveInteger“/>  
</owl:DatatypeProperty>
```

Literal



- Instance declaration

```
<Professor rdf:about=„Sokrates“/>
```

- Instance definition (details)

```
<Professor rdf:about=„Sokrates“>
```

```
  <teaches rdf:resource=„Logic“/>
```

```
  <hasPersonalNumber
```

```
    rdf:datatype=„&xsd;positiveInteger“
```

```
    >2125</hasPersonalNumber>
```

```
</Professor>
```

- Instance equivalence

```
<rdf:Description rdf:about =„Sokrates“>
```

```
  <owl:sameAs rdf:resource=„Socrates“/>
```

```
</rdf:Description>
```

- **owl:priorVersion**
 - ◆ indicates earlier versions of the current ontology
 - ◆ used for ontology management, but no formal meaning
- **owl:versionInfo**
 - ◆ contains information about the current version, e.g. keywords
- **owl:backwardCompatibleWith**
 - ◆ contains a reference to another ontology (e.g. previous version)
 - ◆ all identifiers from the previous version have the same intended interpretations in the new version
- **owl:incompatibleWith**
 - ◆ indicates lack of compatibility with specific ontology
 - ◆ mostly used to indicate major change in ontology version, where documents created with previous version of this ontology will not be correctly interpreted in this version (lack of backward compatibility)

- owl:ObjectProperty
- owl:DatatypeProperty
- owl:inverseOf
- owl:TransitiveProperty
- owl:SymmetricProperty
- owl:FunctionalProperty
- owl:InverseFunctionalProperty

▪ owl:ObjectProperty

- ◆ Defines relations between instances of two classes
- ◆ Both subject and object in statement with object property have to be instances of specific class(es) – **both must have URIs**

```
<owl:ObjectProperty rdf:about=„teaches“>  
  <rdfs:domain rdf:resource=„Professor“/>  
  <rdfs:range rdf:resource=„Lecture“/>  
</owl:ObjectProperty>
```

▪ owl:DatatypeProperty

- ◆ Defines relations between instances of classes and RDF literals and XML Schema datatypes
- ◆ Subject of the statement is a resource, but object is a **literal**

```
<owl:DatatypeProperty rdf:about=„hasPersonalNumber“>  
  <rdfs:domain rdf:resource=„Professor“/>  
  <rdfs:range rdf:resource=„&xsd;positiveInteger“/>  
</owl:DatatypeProperty>
```

▪ owl:inverseOf

- ◆ Explicitly defines property as an inverse of another property in the ontology

```
<owl:ObjectProperty rdf:ID="teaches">  
  <rdfs:range rdf:resource="#Lecture"/>  
  <rdfs:domain rdf:resource="#Professor"/>  
  <owl:inverseOf rdf:resource="#isTaughtBy"/>  
</owl:ObjectProperty>
```

- ◆ Reasoner can use it to infer new knowledge
- ◆ If the knowledge base contains
 - Staab -[teaches] → SemanticWeb
- ◆ Reasoner will infer
 - SemanticWeb -[isTaughtBy] → Staab

▪ owl:TransitiveProperty

- ◆ For property p defined as transitive, if $p(x,y)$ and $p(y,z)$, then also $p(x,z)$

```
<owl:TransitiveProperty rdf:ID="ancestorOf">  
  <rdfs:range rdf:resource="#Person" />  
  <rdfs:domain rdf:resource="#Person" />  
</owl:TransitiveProperty>
```

- ◆ If the knowledge base contains

- Luisa -[ancestorOf] → Barbara
- Barbara -[ancestorOf] → Agatha

- ◆ Reasoner will infer

- Luisa -[ancestorOf] → Agatha

▪ owl:SymmetricProperty

- ◆ Symmetric property p states that if pair of resources x and y are in relationship $p(x,y)$, this relationship also is between y and x : $p(y, x)$

```
<owl:SymmetricProperty rdf:ID="friendOf">  
  <rdfs:range rdf:resource="#Person"/>  
  <rdfs:domain rdf:resource="#Person"/>  
</owl:SymmetricProperty>
```

- ◆ If the knowledge base contains

- Robert -[friendOf] → Paul

- ◆ Reasoner will infer

- Paul -[friendOf] → Robert

▪ owl:FunctionalProperty

- ◆ Relationship is a FunctionalProperty, when it has no more than **one value for each individual** (it may have no values for an individual)

- ◆ It can be described as having a unique property

```
<owl:FunctionalProperty rdf:ID="hasMother">  
  <rdfs:range rdf:resource="#Person" />  
  <rdfs:domain rdf:resource="#Person" />  
</owl:FunctionalProperty>
```

- ◆ If the knowledge base contains

- Robert -[hasMother] → Barbara

- ◆ It means that Robert can have only **one** mother

- ◆ If there is another statement about Robert having second mother, it violates the functionality of the property

▪ owl:InverseFunctionalProperty

- ◆ If a property is inverse functional then the inverse of the property is functional.
- ◆ The inverse of the property has at most one value for each individual.

```
<owl:InverseFunctionalProperty rdf:ID="hasTaxNo" >  
  <rdfs:domain rdf:resource="#Person" />  
  <rdfs:range rdf:resource="#TaxReference" />  
</owl:InverseFunctionalProperty>
```

- ◆ In other words: if you know subject of the statement, you will find exactly one object. And vice-versa.
 - For one person there is only one tax number
 - For one tax number there is only one person

- We can specify minimum and maximum number using **owl:minCardinality** and **owl:maxCardinality**
- It is possible to specify a precise number by using the same minimum and maximum number
- For convenience, OWL offers also **owl:cardinality**

```
<owl:Class rdf:about="#Course">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isTaughtBy" />
      <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">
        1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

No Unique-Names Assumption

- OWL does not support the unique-names assumption
 - ◆ Different names or IDs of two resources do not necessarily imply that these are two different resources

- Example:
 - ◆ There should be only one president of a country
... but ...
 - ◆ knowledge base contains two different entries about country president:
 - X is president of Neverland
 - Y is president of Neverland

 - ◆ OWL reasoner should not report an error
 - ◆ Instead ... it infers that resources X and Y are equal

OWL DL: Restrictions

- Vocabulary partitioning
Any resource is allowed to be only
 - ◆ a class,
 - ◆ a data type,
 - ◆ a data type property,
 - ◆ an object property,
 - ◆ an individual,
 - ◆ a data value,
 - ◆ or part of the built-in vocabulary,
 - ◆ **and not more than one of these**

- Explicit typing
 - ◆ The partitioning of all resources must be stated explicitly

OWL DL: Restrictions (2)

- Property Separation
 - ◆ The set of **object properties** and **datatype properties** are disjoint
- Following types of properties can **only** be object properties:
 - `owl:inverseOf`
 - `owl:FunctionalProperty`
 - `owl:InverseFunctionalProperty`
 - `owl:SymmetricProperty`
- Cardinality restrictions are not allowed for transitive properties
- Anonymous classes are only allowed to be defined as:
 - ◆ domain and range of **owl:equivalentClass** or **owl:disjointWith**
 - ◆ range (only!) of **rdfs:subClassOf**

Summary

- OWL is the proposed standard for Web ontologies by W3C
 - ◆ Has defined vocabulary with specific semantics
- OWL comes in three flavors
 - ◆ OWL Full, **OWL DL** and OWL Lite
- OWL builds upon RDF and RDFS
- Formal semantics and reasoning support is provided through the mapping of OWL on logics
 - ◆ Predicate logic and **description logics** have been used for this purpose
- OWL DL permits efficient reasoning support

End of episode 1 → More OWL details in next lecture