OWL
Ontology Web Language
(part 2)

Maciej Janik

Semantic Web
2009-07-03
Outline

- Relation to Description Logic
- Reasoning in open and closed world
- OWL constructions
  - Equality, inequality, disjointness …
  - Restrictions on properties
  - Boolean combination of classes
  - Enumeration
OWL DL and Description Logic

- **AL** - Attributive Language.
  This is the base language which allows:
  - Atomic negation (negation of concepts that do not appear on the left hand side of axioms)
  - Concept intersection
  - Universal restrictions
  - Limited existential quantification

- **C** – Complex concept negation

- **ALC_{R+}** – Attributive Language, with complex concept negation and transitive roles
OWL DL and Description Logic

OWL DL is very close to the SHOIN(D) Description Logic

- **S** = ALC_{R^+}
- **H** — role Hierarchies
  - rdfs:subPropertyOf
- **O** — nOminals
  - enumerated classes of object value restrictions - owl:oneOf, owl:hasValue
- **I** — Inverse roles
  - owl:inverseProperty, owl:FunctionalInverseProperty
- **N** — simple Number restrictions
  - concept constructors n≤R and n≥R, owl:Cardinality, owl:MaxCardinality
- **(D)** — Datatypes (datatype properties)
### OWL DL and Description Logic

#### Concepts

<table>
<thead>
<tr>
<th>ALC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td>$A, B$</td>
</tr>
<tr>
<td>Not</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>And</td>
<td>$C \sqcap D$</td>
</tr>
<tr>
<td>Or</td>
<td>$C \sqcup D$</td>
</tr>
<tr>
<td>Exists</td>
<td>$\exists R.C$</td>
</tr>
<tr>
<td>For all</td>
<td>$\forall R.C$</td>
</tr>
</tbody>
</table>

#### Ontology (=Knowledge Base)

**Concept Axioms (TBox)**

- Subclass: $C \sqsubseteq D$
- Equivalent: $C \equiv D$

**Role Axioms (RBox)**

- Subrole: $R \sqsubseteq S$
- Transitivity: $\text{Trans}(S)$

**assertional Axioms (ABox)**

- Instance: $C(a)$
- Role: $R(a,b)$
- Same: $a = b$
- Different: $a \neq b$

### Roles

<table>
<thead>
<tr>
<th>Description</th>
<th>$R$</th>
<th>$R^{-}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inverse</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### $S = \text{ALC} + \text{Transitivity}$

**OWL DL = SHQIN(D)** (D: concrete domain)
Open and closed world assumption

- **Open world assumption**
  - If there is no information in knowledge base that something is not true, we cannot say that it is false.
  - If we do not know about some fact, we simply do not know and do not assume any knowledge about it.

- **Closed world assumption**
  - If there is no information in knowledge base that something is false, we can say that it is true – it will not cause a contradiction.

- **Problem:** it is possible to derive false statements from the knowledge base with closed world assumption.
Open and closed world example

- **Question:** “Was there a flood lately here?”

- **Open world answer:** “I do not know if there was lately flood here, but this information is not sufficient to conclude that it haven’t been."

- **Closed world answer:** “I do not know if there was lately flood here, if it was I should have heard about it, so I conclude there was no flood lately here.”

- OWL uses **open world assumption**
OWL Classes

- OWL defines most general and most specific class

- `owl:Thing` is the most general class, which contains everything

- `owl:Nothing` is the empty class

\[
\text{Thing} = \text{Nothing} \cup \text{Nothing}
\]

\[
\text{Nothing} = \text{Thing} = \text{Nothing} \cup \text{Nothing} = \text{Nothing} \cap \text{Nothing} = \emptyset
\]

- `owl:equivalentClass` defines equivalence of classes

```xml
<owl:Class rdf:ID="faculty">
  <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```
Equality and Inequality

- Equality and inequality can be stated between arbitrary resources in OWL
  - Equality or inequality of classes can only be expressed in OWL Full

- Properties
  - owl:sameIndividualAs,
  - owl:sameAs
  - owl:differentFrom
Equality and Inequality

```xml
<rdf:Property rdf:ID="sameIndividualAs">
    <rdfs:domain rdf:resource="#Thing"/>
    <rdfs:range rdf:resource="#Thing"/>
</rdf:Property>

<rdf:Property rdf:ID="sameAs">
    <EquivalentProperty rdf:resource="#sameIndividualAs"/>
</rdf:Property>

Example

<person about="#Bill_Gates">
    <owl:sameAs>
        <person about="#William_Henry_Gates_III">
        </person>
    </owl:sameAs>
</person>
```
Distinct Objects

- Unique, different URIs in OWL **do not guarantee** that entities are different.

- Two entities with different URIs, can be mapped to each other explicitly (owl:sameAs) or implicitly (by using functional or inverse functional property).

- To ensure that different individuals are indeed recognized as such, we must explicitly express their inequality:

```
<course rdf:about="#12345">
  <owl:differentFrom rdf:resource="12367"/>
</course>
```
Distinct Objects

- OWL provides a compact notation to express inequality of all individuals in a given collection

```xml
<owl:allDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <course rdf:about="12345"/>
    <course rdf:about="12367"/>
    <course rdf:about="12389"/>
  </owl:distinctMembers>
</owl:allDifferent>
```
Property Restrictions

- For a specific class and property, it is possible in OWL to define further restrictions
  - combination of classes allowed as domain/range
  - individuals allowed for this specific class and relationship

- It is possible to use previously defined relationship (in general context) and customize specific restrictions for this property for specific class

- Effectively it defines some class C, which describes all objects that satisfy the given restrictions
  - C can remain anonymous
Property Restrictions

- Restriction class is defined using `owl:Restriction`
- Restriction for specific property uses `owl:onProperty` element and includes one or more restriction declarations
- Cardinality restrictions is a special type of restriction (at least 1, exactly 4, at most 3,...)

- Restrictions can be defined using following properties
  - `owl:allValuesFrom`
    - universal quantification
  - `owl:hasValue`
    - specific value
  - `owl:someValuesFrom`
    - existential quantification
Restrictions: owl:allValuesFrom

```xml
<owl:Class rdf:about="#firstYearCourse">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isTaughtBy"/>
      <owl:allValuesFrom rdf:resource="#Professor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Restricting that only Professor can teach courses on the first year. Still uses previously defined property “isTaughtBy” – do not introduce new relationships in ontology.
Restricting teaching of a specific class of courses to a given individual. Here: only Steffen Staab can teach any course that belongs to a class of semantic web.

```xml
<owl:Class rdf:about="#SemanticWebCourse">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isTaughtBy"/>
      <owl:hasValue rdf:resource="#Steffen_Staab"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
Restricting that PhD student is allowed to teach only seminar courses.
Narrowing semantics of property “teaches” to “seminar” courses for this specific class “PHDstudent”
**Boolean combination of classes**

- OLW allows to combine classes in similar way as sets in set theory

- Supported operations
  - Complement (A)
  - Union (A and B)
  - Intersection (A and B)
<owl:Class rdf:about="#SpamMessage">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:complementOf rdf:resource="#RealEmailMessage"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

Meaning: everything except given class (or combination of classes)
The new class (peopleAtUniversity) is a union of classes – it is not defined as subclass of such union.

```xml
<owl:Class rdf:ID="peopleAtUniversity">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#staffMember"/>
    <owl:Class rdf:about="#student"/>
  </owl:unionOf>
</owl:Class>
```
This is a definition of a new class by specific restriction of a workplace. If a person teaches at computer science department and is also a faculty, she or he is a faculty at computer science.
Nesting of Boolean Operators

```xml
<owl:Class rdf:ID="adminStaff">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#staffMember"/>
    <owl:Class>
      <owl:complementOf>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#faculty"/>
            <owl:Class rdf:about="#techSupportStaff"/>
          </owl:unionOf>
        </owl:Class>
      </owl:complementOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```
Class Disjointness

Specifying that two classes cannot have common subclasses or individuals

```
<rdf:Property rdf:ID="disjointWith">
   <rdfs:label>disjointWith</rdfs:label>
   <rdfs:domain rdf:resource="#Class"/>
   <rdfs:range rdf:resource="#Class"/>
</rdf:Property>
```

Example:

```
<owl:Class rdf:ID="redWine">
   <owl:disjointWith>
      <owl:Class rdf:about="#whiteWine"/>
   </owl:disjointWith>
</owl:Class>
```
<owl:Class rdf:ID="weekdays">
    <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Monday"/>
        <owl:Thing rdf:about="#Tuesday"/>
        <owl:Thing rdf:about="#Wednesday"/>
        <owl:Thing rdf:about="#Thursday"/>
        <owl:Thing rdf:about="#Friday"/>
        <owl:Thing rdf:about="#Saturday"/>
        <owl:Thing rdf:about="#Sunday"/>
    </owl:oneOf>
</owl:Class>
To remember …

- OWL offers rich vocabulary, semantics and expressiveness

- OWL DL is mapped to Description Logic
  - Efficient reasoning is enabled
  - Constructions supported in OWL can be directly expressed in DL

- Modeling in OWL is not trivial
  - Variety of different constructions to express the same intent
  - Use of complex constructions can imply some facts that we haven’t foreseen