Laboratory for Applied Ontology

Institute of Cognitive Science and Technology
Italian National Research Council

Professional master on technologies for e-government

Lecture 11-12:

Basics for implementing OWL ontologies in Protégé

ALESSANDRO OLTRAMARI

oltramari@loa-cnr.it

Laboratory for Applied Ontology (CNR), Trento, Italy:

http://www.loa-cnr.it

Outline

LECTURE 11 (today):

- Ontology Web Language (OWL): main features and examples
  - Questions?
- *Protégé*: an integrated platform for developing ontologies

LECTURE 12 (next Wednesday)

- Questions?
- Case study: an implemented ontology for e-government

LECTURE 11

Ontology Languages

- Ontology languages allow users to write explicit **formal specifications** of **conceptualizations** related to a given domain.
  - RDF, RDF(S), DAML-ONT, OIL, DAML+OIL
  - OWL 1.0 (W3C'standard)

- Core requirements:
  - Well-defined syntax
  - Well-defined semantics
  - Efficient reasoning support
  - Sufficient expressive power

Syntax

- Well-defined syntax is important for enabling machine-processing of information

- RDF syntax (XML-based) is hard to be considered as user-friendly!

Instead of directly writing the code, users exploits ontology development tools, such as Protégé, Swoop, Top Braid Composer,…

Semantics

- Well-defined (formal) semantics describes *rigorously* the meaning of knowledge
- It allows for precise reasoning. For ontological knowledge, we can reason about
  - **Class membership**: if *x* is an instance of class C, and C is a subclass of D, then we can infer that *x* is an instance of D
  - **Equivalence of classes**: If class A is equivalent to class B, and class B equivalent to class C, then A is equivalent to C, too.
  - **Consistency**: given *x* instance of class A and suppose that:
    - A is a subclass of B ∩ C
    - A is subclass of D
    - B and D are disjoint

    then we have an inconsistency → the ontology must be fixed
  - **Classification (or subsumption reasoning)**: if we have declared that certain property/value pairs are sufficient condition for membership of a class A, and if *x* satisfies these conditions, we can conclude that x must be instance of A.

Efficient reasoning support

- Inferences like the one presented before can be made mechanically, instead of manually.

- Reasoning support is important for
  - Check the consistency of an ontology
  - Check for unintended relationships between classes
  - Automatically classify instances in classes

- Automatic reasoning help for
  - Designing large ontologies (eventually involving multiple authors)
  - Integrating and sharing different ontologies

Ontology Web Language (OWL)

- Formal semantics and reasoning support are provided by mapping ontology language to logical formalisms and using automated 'reasoners' that already exist for those formalisms.
- OWL is (partially – we'll see why) mapped on a description logic, and makes use of existing reasoners such as FaCT and RACER.
- Description logics are a **decidable** fragment of FOL.

REMINDER: Logics are decidable if computations/algorithms based on the logic will terminate in a finite time.

OWL sub-languages

- **OWL-LITE**

  Syntactically simplest →used for class hierarchy and simple constraints
- **OWL-DL**

  Based on DL → used when automated reasoning is needed
- **OWL-Full**

  Most expressive →used when expressivity is more important than reasoning

Basic components of OWL ontologies

Class

- In the following we introduce some examples of OWL core components; most are taken from the ontology pizza.owl (we'll see how this ontology looks like in the Protégé-OWL platform).

```
<owl:Class rdf:ID="AnchoviesTopping">
  <rdfs:subClassOf rdf:resource="#FishTopping"/>
  <owl:disjointWith rdf:resource="#PrawnsTopping"/>
  <owl:disjointWith rdf:resource="#MixedSeafoodTopping"/>
</owl:Class>
```

- OWL extends RDF: it adopts RDF meaning of classes and properties, extending the expressivity with its own primitives
- `owl:Thing` → the most general class (it subsumes every class);
- `owl:Nothing` → the empty class (every class subsumes it)

Object Property

- Object property relates classes (objects) to other classes (objects):

```
<owl:ObjectProperty rdf:about="#hasTopping">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasIngredient"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Pizza"/>
  <rdfs:range rdf:resource="#PizzaTopping"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="isToppingOf"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

- `InverseOf` interchange domain with range

Datatype Property

- Datatype property relates classes (objects) to datatype values:

```
<owl:DataProperty rdf:ID="#Table_number">
   <rdfs:range rdf:resource=

   "http://www.w3.org/2001/XMLschema#nonNegativeInteger"/
   >
  </owl:DataProperty>
```

- OWL does not provide predefined datatypes: it uses XML Schema datatypes
- Language layering: **XML→XML(S)→RDF→RDF(S)→OWL**

Property restrictions (1/2)

- `<rdfs:subClassOf>` relates a class C with an higher class C'.
- OWL allows to declare that if C satisfies some conditions, which namely represent all the conditions a sub-class of C' must have, then C is subclass of C'.

```
<owl:Class rdf:ID="AmericanHotPizza">
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="#hasTopping"/>
       <owl:someValuesFrom
            rdf:resource="#JalapenoPepperTopping"/>
     </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

- **existential quantification (∃) =** `owl:someValuesFrom`
- **universal quantification (∀) =** `owl:AllValuesFrom`

Property restrictions (2/2)

- **Other types of restrictions are**
  - `owl:hasValue` **(states the specific value that a property must have)**
  - `owl:minCardinality` **(states the minimum value that a property must have)**
  - `owl:maxCardinality` **(states the maximum value that a property must have)**

- `owl:Restriction` **defines an anonymous class, which has no id, no** `owl:Class` **definitions, and local scope; it can only be used in the one place where the restriction appears.**

```
<owl:Class rdf:ID="PizzaTopping">
     <rdfs:subClassOf>
        <owl:Restriction>
           <owl:onProperty rdf:resource="#hasTopping"/>
           <owl:minCardinality
                    rdf:datatype ="&xsd;nonNegativeInteger>1
           </owl:minCardinality>
       </owl:Restriction>
     </rdfs:subClassOf
</owl:Class>
```

Special properties

- `owl:TransitiveProperty`
  - e.g. “is taller than”
- `owl:SymmetricProperty`
  - e.g., “is sibling of”
- `owl:FunctionalProperty` : defines a property that has at most one unique value for each object
  - e.g. “age”; “height”,…
- `owl:InverseFunctionalProperty` : defines a property for which two different objects cannot have the same value
  - e.g. “isTheIdentityCardNumberOf”

Boolean combinations

- owl:complement owl:unionOf owl:intersectionOf
- **CheesyPizza is equivalent to the intersection between Pizza class and the class of pizzas that have some cheese topping**

```
<owl:Class rdf:ID="CheeseyPizza">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:someValuesFrom rdf:resource="#CheeseTopping"/>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#hasTopping"/>
            </owl:onProperty>
          </owl:Restriction>
          <owl:Class rdf:about="#Pizza"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
```

Enumerations

```
<owl:oneOf rdf:parseType="Collection">
   <Country rdf:ID="America"/>
   <Country rdf:ID="England"/>
   <Country rdf:ID="France"/>
   <Country rdf:ID="Germany"/>
   <Country rdf:about="#Italy"/>
</owl:oneOf>
```

References

- G. Antoniou and F. van Harmelen, “Web Ontology Language”, Handbook on Ontologies, Springer, 2004.
- M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe, “A practical guide to building OWL ontologies using The Protégé-OWL plug-in and CO-ODE Tools Edition 1.0. Available at: http://protege.stanford.edu/
- http://protege.stanford.edu/plugins/owl/publications/2004-07-06-OWL-Tutorial.ppt
- http://www.w3.org/TR/owl-features/
- http://www.sts.tu-harburg.de/r.f.moeller/racer/
- http://www.mindswap.org/2003/pellet/
- http://www.cs.man.ac.uk/ horrocks/FaCT/