An Introduction to OWL and its Alternatives

Nigam Shah
nigam@stanford.edu

NATIONAL CENTER FOR
BIOMEDICAL ONTOLOGY
**OWL**

- **Web Ontology Language**
- Recommended by W3C since Feb 2004
- Based on predecessors (DAML+OIL)
- A Web Language: Based on RDF(S)
- An Ontology Language: Based on logic
- Three varieties
  - OWL-full
  - OWL-DL ("OWL")
  - OWL-Lite
The Three Sublanguages of OWL

**OWL Full**
- Maximum expressiveness with syntactic freedom of RDF with no computational guarantees

**OWL DL**
- Highly expressive while retaining computational completeness

**OWL Lite**
- Classification hierarchy and simple constraints
Working with OWL syntax is not easy

```xml
<owl:Class rdf:ID="Virus">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    </rdfs:comment>
  <owl:disjointWith>
    <owl:Class rdf:ID="Bacterium"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="MicroOrganism"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Virus</rdfs:label>
  </owl:Class>
  <owl:Class rdf:about="#Bacterium">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#MicroOrganism"/>
    </rdfs:subClassOf>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      Bacterium</rdfs:label>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      </rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#MicroOrganism">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:ID="Organism"/>
          <owl:Restriction>
            <owl:someValuesFrom>
              <owl:Class rdf:ID="MicroScale"/>
            </owl:someValuesFrom>
          </owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:ID="asScaleRealm"/>
          </owl:onProperty>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
```
Tools are being developed for OWL

Even with nice XML tools, RDF syntax is not very nice to work with
## Alternatives?

<table>
<thead>
<tr>
<th>Logical formalism</th>
<th>Reasoners</th>
<th>Tools to “speak” the language</th>
<th>Size of the user community</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL</td>
<td>Description Logic</td>
<td>Fact++, Pellet, Racer</td>
<td>Protégé, Swoop</td>
<td>~ in thousands</td>
</tr>
<tr>
<td>OBO</td>
<td>DL-compatible for now</td>
<td>OBO-edit reasoner</td>
<td>OBO-edit</td>
<td>~ in hundreds</td>
</tr>
<tr>
<td>Frames (GFP) &amp; OKBC</td>
<td>?</td>
<td>Ocelot, Ontolingua, GKB-Editor</td>
<td>~ in hundreds</td>
<td>AI community standard</td>
</tr>
<tr>
<td>KIF</td>
<td>expression of arbitrary logical sentences</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Loom</td>
<td>Not DL</td>
<td>Loom “classifier”</td>
<td>Loom</td>
<td>Small</td>
</tr>
<tr>
<td>XOL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHOE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OML</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDF(S)</td>
<td>Subsumed by OWL</td>
<td>Subsumed by OWL</td>
<td>Subsumed by OWL</td>
<td>Subsumed by OWL</td>
</tr>
<tr>
<td>DAML + OIL</td>
<td>Subsumed by OWL</td>
<td>Subsumed by OWL</td>
<td>Subsumed by OWL</td>
<td>Subsumed by OWL</td>
</tr>
</tbody>
</table>
Basic Protégé-OWL usage

Nigam Shah
nigam@stanford.edu
Protégé OWL: a GUI environment

- OWL environment within PROTÉGÉ framework
- Most widely used tool for editing and managing OWL ontologies
- Approx 75,000 registered users
Protégé OWL features

• Loading and saving OWL files & databases
• Graphical editors for class expressions
• Access to description logics (DL) reasoners via Protégé GUI and the DIG interface
• Ontology visualization plug-ins
• Built on Protégé platform
  • Can hook in custom-tailored components
  • Can serve as API for new applications (including web applications)
PROJECTS
Loading OWL files

1. If you only have an OWL file:
   - File ➔ New Project
   - Select OWL Files as the type
   - Tick Create from existing sources
   - Next to select the .owl file

2. If you’ve got a valid project file*:
   - File ➔ Open Project
   - select the .pprj file

* ie one created on this version of Protégé - the s/w gets updated once every few days, so don’t count on it unless you’ve created it recently– safest to build from the .owl file if in doubt
(Create or load an OWL project)

File → New Project

OR

File → Open Project
Protégé OWL Overview

- **Classes**
  - Subclass relationships
  - Disjoint classes

- **Properties**
  - Characteristics (transitive, inverse)
  - Range and Domain
  - ObjectProperties (references)
  - DatatypeProperties (simple values)

- **Individuals**
  - Property values

- **Class Descriptions**
  - Restrictions
  - Logical expressions

OWL for data exchange

OWL for classification and reasoning
Ontology Development Process

In reality - an iterative process:

determine scope → consider reuse → enumerate terms → define classes → define properties → define constraints → create instances

define properties → define classes → define properties → define constraints → create instances → define classes → create instances

consider reuse → define properties → define constraints → create instances
Establish Purpose

What will the ontology be used for?

Classification of Pneumonia:

- Bacterial Pneumonia (caused by bacteria)
- Pneumococcal Pneumonia (caused by a particular kind of bacteria)
- Viral Pneumonia (caused by viruses)
- Mixed Pneumonia (caused by both bacteria and viruses)
Enumerate Important Concepts

- What are the terms we need to talk about? *Pneumonias, infectious organisms.*
- What are the properties of these terms? `hasRadiologyFinding, hasLocus, hasCause`.
- What do we want to say about the terms? *Pneumonias cause radiology opacity findings*
  *Pneumonias are located in lung*
  *Mixed pneumonias are caused by bacteria and viruses.*
  ...

---

determine scope  consider reuse  enumerate terms  define classes  define properties  define constraints  create instances
CLASSES
Classes

- Sets of **individuals** with common characteristics
- **Individuals** are instances of at least one class

City
  - Sydney
  - Cairns

Beach
  - BondiBeach
  - CurrawongBeach
Superclass Relationships

- Classes organized in a hierarchy implies subsumption
- Direct instances of subclass are also (indirect) instances of superclasses
Class Relationships

- Classes can overlap arbitrarily
- Classes are assumed non-disjoint by default (i.e., they may share instances)
Class Disjointness

- All classes could potentially overlap
- Specify **disjointness** to make sure they don’t share instances
Class Editor

Class annotations (for class metadata)

Class name and documentation

Properties “available” to Class

Disjoints widget

Conditions Widget

Class-specific tools (find usage etc)
Define classes and the class hierarchy

- Identify Classes (from the previous term list)
  - If something can have a kind then it is a Class
  - “Kind of Pneumonia” V - Pneumonia is a Class
  - “Kind of Samson” X - Samson is an individual
  - “Kind of Bacteria” V Bacteria is a Class
Define classes and the class hierarchy

Arrange Classes in an hierarchy
PneumococcalPneumonia is a subclass of Pneumonia
   Every PneumococcalPneumonia is a Pneumonia
Pneumococcus is a subclass of Bacteria
   Every Pneumococcus is a Bacteria
MixedPneumonia is a subclass of Pneumonia
   Every MixedPneumonia is a Pneumonia
Create classes: “Pneumonia” class
Note that Bacterial Pneumonia
- has superclass Pneumonia as a necessary condition
- Is asserted to be disjoint from its ‘siblings’
What it means

• All BacterialPneumonias are Pneumonias
  • **No BacterialPneumonia** is not a Pneumonia
• Nothing is both:
  • a BacterialPneumonia and a ViralPneumonia
  • a BacterialPneumonia and a MixedPneumonia

**NB:** In OWL classes *can overlap* unless declared disjoint!
Add Annotations on Classes
Another Way to Create Classes

- A class can be the **union** of two classes
  - An InfectiousPneumonia is either a BacterialPneumonia or a ViralPneumonia
- A class can be the **intersection** of two classes
  - A MixedPneumonia is any Pneumonia that is caused by both Bacteria and Viruses
- A class can be the **complement** of another class
  - Noninfectious pneumonia is any pneumonia that is not caused by an infectious agent (bacteria or virus)
An InfectiousPneumonia is a Pneumonia that is either a BacterialPneumonia or a ViralPneumonia.
PROPERTIES
OWL Properties

- **Datatype Property** – relates Individuals to data (int, string, float etc)
  - Pneumonia hasRadiologyFinding xsd:String

- **Object Property** – relates Individuals
  - BacterialPneumonia hasCause Bacterium

- **Annotation Property** – for attaching metadata to classes, individuals or properties
  - OntologyClass hasAuthor Natasha
Datatype Properties

- Link individuals to primitive values (integers, floats, strings, booleans etc)
- Often: AnnotationProperties without formal “meaning”

<table>
<thead>
<tr>
<th>Sydney</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasSize = 4,500,000</td>
</tr>
<tr>
<td>isCapital = true</td>
</tr>
<tr>
<td>rdfs:comment = “Don’t miss the opera house”</td>
</tr>
</tbody>
</table>
Object Properties

- Link two individuals together
- Relationships (0..n, n..m)
Annotation Properties

- To annotate classes, properties, and individuals
- Usually used for documentation

Sydney

```rdfs:comment```
My comment

```
hasAuthor```
Kaustubh Supekar
Properties of an OWL property

- Functional
  - Person has_Mother Mother

- Transitive
  - A hasPart B, B hasPart C ==> A hasPart C

- InverseFunctional
  - Person has_SSN SSN

- Symmetric
  - A worksWith B ==> B worksWith A
Define Properties of Classes

- Properties in a class definition describe attributes of instances of the class and relations to other instances
  - Each Pneumonia will have radiology findings and a cause
  - Each cause for pneumonia will have a causative organism.
Create object property “has_part”

• Click on properties tab
• Click on Create_Object_property icon and create has_part
Object property hasLocus (already present)
Create New Datatype Property, “hasRadiologyFinding”

Datatype = string
Create annotation property “hasAuthor”
RESTRICTIONS
Restrictions (Overview)

• An **anonymous class** consisting of all individuals that fulfill the condition

• Define a condition for property values
  - allValuesFrom
  - someValuesFrom
  - hasValue
  - minCardinality
  - maxCardinality
  - cardinality
Define Constraints : OWL Restrictions

- **Quantifier restriction**
  - How to represent the fact that every pneumonia must be located in a lung?

- **Cardinality restrictions**
  - How to represent that a Hand must have 5 fingers as parts?

- **hasValue restrictions**
  - How to define the value of a relation for a class? (relationship between class and an individual)
Quantifier Restrictions

Restrictions are of the form

\[ \text{All members of class C have as values for property p} \]
\[ \text{some things of Class D (∃)} \]
\[ \text{only things of class D (∀)} \]
\[ \text{at least | at most | exactly n things} \]

Examples

• "some" (someValuesFrom) (∃) (Existential)
  Cheesy_Pizza has_base someValuesFrom Cheese_Topping.
  Implies- “All cheesy pizzas have some (at least 1) topping that is a cheesey topping”

• "only" (allValuesFrom) (∀) (Universal)
  VegetarianPizza has_topping allValuesFrom Vegetarian_Topping.
  Implies - “All Vegetarian pizzas have only toppings that are Vegetarian Toppings”
Creating Restrictions

Restricted Property
Filler Expression
Expression Construct Palette
Syntax check
Restriction Type
Create a restriction: Add a datatype property

“All pneumonias are disorders that have a radiological finding of opacification”
"All pneumonias are disorders that are located in some lung and have a radiological finding of opacification"
Add more object properties

- BacterialPneumonia is caused by some bacteria
  - BacterialPneumonia $\sqsubseteq$ causedBy some Bacteria
  - BacterialPneumonia $\rightarrow$ $\exists$ causedBy.Bacteria
- ViralPneumonia is caused by some virus
  - ViralPneumonia $\sqsubseteq$ causedBy some Virus
- MixedPneumonia is caused by some bacteria and by some virus
  - MixedPneumonia $\sqsubseteq$ (causedBy some Bacteria) $\sqcap$ (causedBy some Virus)
Using expression editor

“All Mixed Pneumonias are Pneumonias caused by Bacteria or by Viruses”
Class Descriptions

• Define the “meaning” of classes
• Description Logic expressions ("anonymous class expressions") are used:
  • “All national parks have campgrounds.”
  • “A backpackers destination is a destination that has budget accommodation and offers sports or adventure activities.”
• Expressions usually restrict property values
• Reasoners can perform inference/classification
Defined/Primitive Classes

**Necessary Conditions:**
(Primitive / partial classes)
“If we know that something is a X, then it must fulfill the conditions...”

**Necessary & Sufficient Conditions:**
(Defined / complete classes)
“If something fulfills the conditions..., then it is an X.”
**Defined/Primitive Classes**

**Necessary Conditions:** (Primitive classes)
Describes a subclass
“If something is a Class_X, then it must fulfill the conditions...”
*Converse may NOT be true:* “If something fulfills the conditions..., then it is a Class_X.”

**Necessary & Sufficient Conditions:** (Defined classes)
“If something fulfills the conditions..., then it is a Class_X.”
e.g., Disorder is a necessary condition on Pneumonia.

“If something is a Pneumonia, then it is a Disorder”

BUT

“If something is a Disorder, it may not be a Pneumonia”
Necessary & sufficient conditions on BacterialPneumonia

“If N&S conditions, then it is a BacterialPneumonia”

AND

“If something is a BacterialPneumonia, then N&S conditions”
INDIVIDUALS
Individuals

- Represent objects in the domain
- Specific things
- Two names could represent the same “real-world” individual

Sydney

Bondi Beach

Sydneys Olympic Beach
Create instances

Create an instance of a class

• The class becomes a **direct type** of the instance
• Any superclass of the direct type is a **type** of the instance
• Generally, you create instances if you have a “type-of” something
Classification
Reasoners

- Reasoners ("classifiers") infer information that is not explicitly contained within the ontology.
- Standard reasoner services are:
  - **Consistency Checking** (i.e., satisfiability—can a class have any instances?)
  - **Subsumption Checking** (Finding subclasses—is A a subclass of B?)
  - **Equivalence Checking**
  - **Instantiation Checking** (Which classes does an individual belong to?)
- For Protégé we recommend RACER or Fact++ (but other tools with DIG support work too).
- Reasoners can be used at runtime in applications as a querying mechanism.
- Used during development as an ontology "compiler". Ontologies can be compiled to check if the meaning is what was intended.
Run a DL Reasoner with Protégé

**OWL**

- Protégé OWL can work with multiple reasoners
  - Racer (http://www.racer-systems.com/)
  - Pellet (http://www.mindswap.org/2003/pellet/)
  - Fact++ (http://owl.man.ac.uk/factplusplus/)
- Need to install, configure, and run at least one reasoner as a separate process
- Protégé OWL and reasoner exchange information through inter-process communication
An infectious pneumonia is either a bacterial or viral pneumonia.
Now classify...

BacterialPneumonia & ViralPneumonia are now subclasses of InfectiousPneumonia
What does all this mean?

• Description logic (and OWL-DL) provides
  • Expressivity with semantic precision
  • Compositional definitions:
    • define new concepts from old
    • Automatic classification & consistency checking

• Protégé OWL provides a GUI for developing OWL ontologies
Further reading/exploration

• **Protégé**: http://protege.stanford.edu
  • Protégé OWL:
    http://protege.stanford.edu/plugins/owl/
  • Protégé OWL discussion list
  • Protégé Workshops (early 2006)
  • Protégé International Conference

• **OWL tutorial** materials from CO-ODE project site (University of Manchester)
  http://www.co-ode.org/resources/tutorials/

• **NCBO**: http://bioontology.org
More about Protégé OWL

• Documentation on
  http://protege.stanford.edu/plugins/owl/documentation.html

• Excellent tutorial by Mathew Horridge
  http://www.coode.org/resources/tutorials/ProtegeOWLTutorial.pdf

• Other resources at http://www.coode.org/resources/
Acknowledgements

• Daniel Rubin – for providing a great set of Protégé-OWL slides.