Editing OWL Ontologies with Protégé

Jon Atle Gulla

Overview

The Semantic Web and OWL Revisited

Basic OWL

Protege: Classes, Properties

Advanced OWL

Protege: Class Descriptions
The Semantic Web

Shared ontologies help to exchange data and meaning between web-based services

Wine Example Scenario

Tell me what wines I should buy to serve with each course of the following menu.

I recommend Chardonnay or DryRiesling

Books Agent

Grocery Agent

Wine Agent
Ontologies in the Semantic Web

- Provide shared data structures to exchange information between agents
- Can be explicitly used as annotations in web sites
- Can be used for knowledge-based services using other web resources
- Can help to structure knowledge to build domain models (for other purposes)

OWL

- Web Ontology Language
- Official W3C Standard since Feb 2004
- Based on predecessors (DAML+OIL)

- A Web Language: Based on RDF(S)
- An Ontology Language: Based on logic
OWL Ontologies

- What’s inside an OWL ontology
  - Classes + class-hierarchy
  - Properties (Slots) / values
  - Relations between classes (inheritance, disjoints, equivalents)
  - Restrictions on properties (type, cardinality)
  - Characteristics of properties (transitive, …)
  - Annotations
  - Individuals

- Reasoning tasks: classification, consistency checking

OWL Use Cases

- At least two different user groups
  - OWL used as data exchange language (define interfaces of services and agents)
  - OWL used for terminologies or knowledge models

- OWL DL is the subset of OWL (Full) that is optimized for reasoning and knowledge modeling
Protégé OWL Plugin

- Extension of Protégé for handling OWL ontologies
- Project started in April 2003
- Features
  - Loading and saving OWL files & databases
  - Graphical editors for class expressions
  - Access to description logics reasoners
  - Powerful platform for hooking in custom-tailored components

OWL Plugin Architecture

- Protégé OWL API (Logical class definitions, restrictions, etc.)
- Protégé GUI (Expression Editor, Conditions Widget, etc.)
- Protégé OWL GUI (SWRL, OWL-S, etc.)
- Jena API (Parsing, Reasoning)
- OWL File Storage
- DB Storage
- OWL GUI Plugins (SWRL Editors, ezOWL, OWLViz, Wizards, etc.)
Tutorial Scenario

- Semantic Web for Tourism/Traveling
- Goal: Find matching holiday destinations for a customer

I am looking for a comfortable destination with beach access

Tourism Web

Scenario Architecture

- A search problem: Match customer’s expectations with potential destinations
- Required: Web Service that exploits formal information about the available destinations
  - Accommodation (Hotels, B&B, Camping, ...)
  - Activities (Sightseeing, Sports, ...)
Tourism Semantic Web

- Open World:
  - New hotels are being added
  - New activities are offered
- Providers publish their services dynamically
- Standard format / grounding is needed → Tourism Ontology
OWL (in Protégé)

- Individuals (e.g., “FourSeasons”)
- Properties
  - ObjectProperties (references)
  - DatatypeProperties (simple values)
- Classes (e.g., “Hotel”)

Individuals

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

- Sydney
- BondiBeach
- Sydney’s Olympic Beach
ObjectProperties

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

Inverse Properties

- Represent bidirectional relationships
- Adding a value to one property also adds a value to the inverse property
Transitive Properties

- If A is related to B and B is related to C then A is also related to C
- Often used for part-of relationships

DatatypeProperties

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

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

Range and Domain

- Property characteristics
  - Domain: “left side of relation” (Destination)
  - Range: “right side” (Accomodation)
Domains

- Individuals can only take values of properties that have matching domain
  - “Only Destinations can have Accomodations”
- Domain can contain multiple classes
- Domain can be undefined:
  Property can be used everywhere

Superclass Relationships

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

- Classes can overlap arbitrarily

Class Disjointness

- All classes could potentially overlap
- In many cases we want to make sure they don’t share instances
(Create class hierarchy and set disjoints)

![Diagram of class hierarchy and disjoints]

(Create Contact class with datatype properties)

![Diagram of Contact class with datatype properties]
(Edit details of datatype properties)

![Image of editing datatype properties]

(Create an object property hasContact)

![Image of creating an object property]

Jon Atle Gulla
OWL Modeling
(Create an object property with inverse)

(Create the remaining classes and properties)
Class Descriptions

- Classes can be described by their logical characteristics
- Descriptions are “anonymous classes”

Define the “meaning” of classes

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 mostly restrict property values (OWL Restrictions)
Class Descriptions: Why?

- Based on OWL’s Description Logic support
- Formalize intentions and modeling decisions (comparable to test cases)
- Make sure that individuals fulfill conditions
- Tool-supported reasoning

Reasoning with Classes

- Tool support for three types of reasoning exists:
  - Consistency checking:
    Can a class have any instances?
  - Classification:
    Is A a subclass of B?
  - Instance classification:
    Which classes does an individual belong to?
- For Protégé we recommend Pellet
  (but other tools with DIG support work too)
Restrictions (Overview)

- Define a condition for property values
  - allValuesFrom
  - someValuesFrom
  - hasValue
  - minCardinality
  - maxCardinality
  - cardinality
- An anonymous class consisting of all individuals that fulfill the condition

Cardinality Restrictions

- Meaning: The property must have at least/at most/exactly x values
- is the shortcut for
- Example: A FamilyDestination is a Destination that has at least one Accommodation and at least 2 Activities
**allValuesFrom Restrictions**

- **Meaning:** All values of the property must be of a certain type
- **Warning:** Also individuals with no values fulfill this condition (trivial satisfaction)
- **Example:** Hiking is a Sport that is only possible in NationalParks

```
<table>
<thead>
<tr>
<th>Asserted Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>NECESSARY &amp; SUFFICIENT</td>
</tr>
<tr>
<td>NECESSARY</td>
</tr>
<tr>
<td>Sports</td>
</tr>
<tr>
<td>∀ isPossibleIn NationalPark</td>
</tr>
</tbody>
</table>
```

**someValuesFrom Restrictions**

- **Meaning:** At least one value of the property must be of a certain type
- **Others may exist as well**
- **Example:** A NationalPark is a RuralArea that has at least one Campground and offers at least one Hiking opportunity

```
<table>
<thead>
<tr>
<th>Asserted Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>NECESSARY &amp; SUFFICIENT</td>
</tr>
<tr>
<td>NECESSARY</td>
</tr>
<tr>
<td>RuralArea</td>
</tr>
<tr>
<td>Ǝ hasAccomodation Campground</td>
</tr>
<tr>
<td>Ǝ hasActivity Hiking</td>
</tr>
</tbody>
</table>
```
hasValue Restrictions

- Meaning: At least one of the values of the property is a certain value
- Similar to someValuesFrom but with Individuals and primitive values
- Example: A PartOfSydney is a Destination where one of the values of the isPartOf property is Sydney

Enumerated Classes

- Consist of exactly the listed individuals

OneStarRating

TwoStarRating

ThreeStarRating

BudgetAccommodation
Logical Class Definitions

- Define classes out of other classes
  - unionOf (or)
  - intersectionOf (and)
  - complementOf (not)
- Allow arbitrary nesting of class descriptions
  \((A \text{ and } (B \text{ or } C) \text{ and not } D)\)

unionOf

- The class of individuals that belong to class A or class B (or both)
- Example: Adventure or Sports activities

\[ \exists \text{ hasActivity (Sports } \cup \text{ Adventure)} \]
**intersectionOf**

- The class of individuals that belong to both class A **and** class B
- Example: A BudgetHotelDestination is a destination with accommodation that is a budget accommodation **and** a hotel

**Implicit intersectionOf**

- When a class is defined by more than one class description, then it consists of the intersection of the descriptions
- Example: A luxury hotel is a hotel that is also an accommodation with 3 stars
**complementOf**

- The class of all individuals that do not belong to a certain class
- Example: A quiet destination is a destination that is **not** a family destination

**Class Conditions**

**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.”
Class Conditions (2)

NationalPark

QuietDestination

Class Conditions (2)

Class Conditions (2)

Classification

NationalPark

BackpackersDestination

Jon Atle Gulla

OWL Modeling
Classification (2)

- Input: Asserted class definitions
- Output: Inferred subclass relationships

The Manchester OWL Syntax

An abstract readable syntax for OWL ontologies

Standard DL Syntax in OWL

- Definition of VegetarianPizza in Pizza ontology (Protege tutorial)

  ![Diagram of OWL model](image)

- Issues:
  - Difficult to read for non-ontology experts
  - Non-standard symbols

Manchester OWL Syntax

- Logical symbols translated to English
- Infix notation

<table>
<thead>
<tr>
<th>OWL Constructor</th>
<th>DL Syntax</th>
<th>Manchester OWL Syntax Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>C □ D</td>
<td>C AND D</td>
</tr>
<tr>
<td>unionOf</td>
<td>C □ D</td>
<td>C OR D</td>
</tr>
<tr>
<td>complementOf</td>
<td>¬ C</td>
<td>NOT C</td>
</tr>
<tr>
<td>oneOf</td>
<td>{a} ∪ {b}</td>
<td>{a b}</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>∃ R C</td>
<td>R SOME C</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>∀ R C</td>
<td>R ONLY C</td>
</tr>
<tr>
<td>minCardinality</td>
<td>≥ N R</td>
<td>R MIN 3</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>≤ N R</td>
<td>R MAX 3</td>
</tr>
<tr>
<td>cardinality</td>
<td>≈ N R</td>
<td>R EXACTLY 3</td>
</tr>
<tr>
<td>hasValue</td>
<td>⊏ R {a}</td>
<td>R VALUE a</td>
</tr>
</tbody>
</table>

![Diagram of Manchester OWL syntax](image)
Manchester Syntax Examples

- Class of people who have at least one child that has some children that are only men (i.e. Grandparents that only have grandsons)

\[
\text{Person AND} \\hspace{1cm} \text{hasChild SOME (Person AND} \\hspace{1cm} \text{(hasChild ONLY Man) AND} \\hspace{1cm} \text{(hasChild SOME Person))}
\]

\[
\text{Person THAT} \\hspace{1cm} \text{hasChild SOME (Person THAT} \\hspace{1cm} \text{(hasChild ONLY Man) AND} \\hspace{1cm} \text{(hasChild SOME Person))}
\]

Improved Readability of Ontologies

- VegetarianPizza in Protege in Manchester Syntax

- Important macros:
  - ONLYSOME
  - Exclusive OR
Conclusions

- OWL for modeling ontologies
  - Classes, properties and individuals
  - Restrictions
  - Characteristics of properties
  - Reasoning

- Manchester OWL Syntax

Additional Slides

- The following slides explain in some more detail how Protege is used to construct advanced ontological descriptions
(Create an enumerated class out of individuals)

(Create a hasValue restriction)
(Create a hasValue restriction)

(Define a class)

32
(Classify Campground)

(Add restrictions to City and Capital)
(Create defined class BackpackersDestination)

(Create defined class FamilyDestination)
(Create defined class QuietDestination)

(Create defined class RetireeDestination)
(Classification)

(Consistency Checking)
Putting it All Together

- Ontology has been developed
- Published on a dedicated web address
- Ontology provides standard terminology
- Other ontologies can extend it
- Users can instantiate the ontology to provide instances
  - specific hotels
  - specific activities

Ontology Import

- Adds all classes, properties and individuals from an external OWL ontology into your project
- Allows to create individuals, subclasses, or to further restrict imported classes
- Can be used to instantiate an ontology for the Semantic Web
Tourism Semantic Web (2)

Ontology Import with Protégé

- On the Metadata tab:
  - Add namespace, define prefix
  - Check “Imported” and reload your project
Individuals
  <owl:Ontology rdf:about="http://protege.stanford.edu/plugins/owl/owl-library/heli-bunjee.owl#">
    <owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/owl-library/travel.owl#"/>
  </owl:Ontology>

  <owl:Class rdf:ID="HeliBunjeeJumping">
    <rdfs:subClassOf rdf:resource="http://protege.stanford.edu/plugins/owl/owl-library/travel.owl#BunjeeJumping"/>
  </owl:Class>

  <HeliBunjeeJumping rdf:ID="ManicSuperBunjee">
    <travel:isPossibleIn>
      <rdf:Description rdf:about="http://protege.stanford.edu/plugins/owl/owl-library/travel.owl#Sydney">
        <travel:hasActivity rdf:resource="#ManicSuperBunjee"/>
      </rdf:Description>
    </travel:isPossibleIn>
    <travel:hasContact>
      <travel:Contact rdf:ID="MSBInc">
        <travel:hasEmail rdf:resource="msb@manicsuperbunjee.com"/>
        <travel:hasCity rdf:resource="Sydney"/>
        <travel:hasStreet rdf:resource="Queen Victoria St"/>
        <travel:hasZipCode rdf:resource="1240"/>
      </travel:Contact>
    </travel:hasContact>
    <rdfs:comment rdf:resource="#ManicSuperBunjee">Manic super bunjee now offers nerve wrecking jumps from 300 feet right out of a helicopter. Satisfaction guaranteed.</rdfs:comment>
  </HeliBunjeeJumping>
</rdf:RDF>