A Practical Introduction to Ontologies and OWL

Context
Ontologies are a critical piece of the Semantic Web jigsaw puzzle, and are already used in various forms to capture knowledge in a machine understandable language. The Web Ontology Language (OWL) is a W3C standard that allows the formalisation of knowledge in a semantically rich model with explicit meaning.

Goals of workshop
This introductory level tutorial is aimed at newcomers to ontologies and those using formalisms other than OWL and will provide attendees with hands-on experience in the construction of ontologies.

Participants will build an OWL ontology using the domain of Pizzas. The aim is to produce a model that can be used in the Manchester PizzaFinder application.

Learning Objectives
Attendees will:

- Understand some of the OWL language elements, and their explicit semantics
- Learn the basic principles of modelling using Description Logic based ontologies
- Use informal modelling techniques to uncover issues relating to knowledge capture
- Gain hands-on experience with ontology development using the Protégé-OWL tools
- Learn how to take advantage of inferencing capabilities to build robust, reusable models
- Be exposed to the growing community of users/developers of OWL

The Pizza Tutorial
The following tutorial is based on pizza ontology courses that have been run at The University of Manchester for a number of years. Pizzas have been chosen as they are fun, well-known, fairly neutral (although we wildly encourage arguing about whether the model is correct), have a relatively small scope and are highly compositional. We are building an ontology to support a menu querying system (something along the lines of the PizzaFinder application (www.co-ode.org/downloads/pizzafinder/)). While building your model, try to keep the application in mind.

Please do a Save As regularly to version your work – it is easy to make mistakes while modelling that are really difficult to track down - we do not want you to lose work.


Session 1: Introduction to Protégé and Primitive Classes in OWL

Starting Protégé and Creating a new project

1. Please start Protégé if you haven’t already – it will be available in your start menu. You will be presented with a startup dialog:

![Startup Dialog](image)

2. Select Create New Project… The Create New Project Wizard appears:

![Create New Project Wizard](image)

3. Select OWL Files (.owl or .rdf)
4. Select Next >
5. Select Next >
6. Select OWL Full
7. Select Finish
8. Protégé is now ready to begin
Exercise 1: Create a Primitive Class Hierarchy

You will now begin to create the primitive classes in your model. This begins with a couple of top-level concepts, including somewhere to put your toppings from the ingredient cards.

1. Create the following tree by using the Create Subclass and Create Sibling Class buttons in the Asserted Hierarchy.

![Image of Asserted Hierarchy]

2. Create further subclasses of your pizza toppings under PizzaTopping, producing a hierarchy based on the groupings you made in exercise 1. We recommend suffixing each topping with ... Topping. Make sure you have at least included the following as they will be used later:

   - MeatTopping
   - CheeseTopping
   - MozzarellaTopping
   - TomatoTopping

   Notice that we are adding a suffix of “Topping” to all the classes – this is to allow for the later possibility of extending our ontology to talk about other types of food such as salads, where Tomato might be different from “Tomato pizza topping”

3. Where required, add additional parents using the Conditions Widget, Add Named Class button. Notice that classes with multiple parents appear in several places in the asserted hierarchy.

4. Create a MeatyVegetableTopping class as a subclass of MeatTopping and VegetableTopping
You should now have a small hierarchy of PizzaToppings. Add to these whenever you have a few minutes as they will allow you to create a larger ontology of pizzas later on.
Exercise 2: Add disjoints

To make sure toppings can not be both meat and vegetable at the same time, add disjoints in to your primitive tree.

You will need to run a reasoner supporting the DIG interface to check the consistency of your ontology.

The reasoner software being used may change from time to time. (check the Protége-OWL webpages to check the currently recommended reasoners).

1. Select one of your top level concepts (eg Pizza)

2. Select Add All Siblings… in the disjoints widget:

3. Select the default Mutually between all siblings in the dialog that appears:

4. Repeat this for each level of your ontology, selecting a class and making all of its siblings disjoint.

5. Once your reasoner is running, periodically classify your ontology by pressing the Classify Taxonomy button on the toolbar:

6. Add more PizzaToppings if you have time, making sure these are disjoint where necessary

Questions:

- Do any of your classes come out as inconsistent? Why?
Exercise 3: Properties and Restrictions
In order to describe our classes we need properties, which are used to relate members of a class. We then add restrictions on the class to state logically how these properties are used.

At this stage we are creating Primitive Classes, which only have Necessary Conditions (in this case, restrictions) on them – these are conditions that must be satisfied by all members of this class.

Step 1 Create Properties

1. Select the Properties Tab
2. Create a new Object Property, hasTopping

Step 2 Create a Pizza

1. Create a new subclass of Pizza called NamedPizza
2. Create a subclass of NamedPizza called MargheritaPizza

Step 3: Create restrictions on MargheritaPizza

1. Select the OWLClasses Tab
2. Select MargheritaPizza
3. Select Create Restriction... on the Conditions Widget
4. In the Restriction dialog that pops up, create a SomeValuesFrom (Existential) restriction along the hasTopping property with a filler of MozzarellaTopping.
Note that by default, restrictions are created as *Necessary Conditions* unless the
*Necessary & Sufficient* heading is selected – for creating primitive classes, only
create Necessary Conditions.

5. Add a further existential restriction on *MargheritaPizza* to state that it
must have at least one topping from *TomatoTopping* .
When entering the filler, you have 2 shortcut methods rather than typing the
entire class name:
- enter a partial name and use *Tab* to autocomplete
- use the *Insert Class…* button on the editor palette (bottom left)

6. Create some other pizzas from the menu and add ingredients in the same
fashion – one restriction per ingredient. Make sure you create at least one
pizza that has some meat on it.

7. If you have not already done it, create a restriction on *Pizza* to state that:
“All pizzas must have at least one base from PizzaBase” You will first need
to create the property *hasBase*
Session 2: Defined Classes and Additional Modelling Constructs in OWL

Exercise 4: Define a MeatyPizza

Creating a defined class is similar to creating a primitive class, but a defined class has one or more Necessary & Sufficient Condition. Classes can easily be migrated between primitive and defined.

We wish to model the statement:

Any Pizza that has at least one topping from MeatTopping is a MeatyPizza

1. Create a new subclass of Pizza called MeatyPizza In general, defined classes are not disjoint from their siblings
2. Create a restriction on MeatyPizza to state that it has at least one topping from MeatTopping
3. Drag both conditions (the new restriction and the superclass, Pizza) up to the Necessary & Sufficient heading. They should move and the class icon should change to contain an equivalence symbol. You can also create restrictions under this heading automatically by selecting it before pressing Create Restriction. The result should be as below:

4. Classify your ontology and check in the Inferred Hierarchy to see what has been classified under the new class. An Inferred Hierarchy panel appears:

All Pizzas that have at least one meat topping should now be subclasses of MeatyPizza.

A classification result panel also appears at the bottom of the screen declaring all inferences that have been made in the current ontology.
Both panels can be shut down using the red cross button.

5. Create several other defined classes of your choosing and classify each time to see if their definition has “captured” the correct named pizzas.
Exercise 5: Define a VegetarianTopping

In order to define a VegetarianPizza, we are going to model the idea of a vegetarian topping. This class is going to be covered by all of the PizzaToppings that are not MeatTopping (or FishTopping/SeafoodTopping or whatever else you have that vegetarians don’t eat). Vegetables, Fruit, Nuts etc are all to be classified as vegetarian toppings.

1. Create a new subclass of PizzaTopping called VegetarianTopping

2. Click on the Necessary & Sufficient heading in the Conditions Widget

3. Click the Create new expression button on the Conditions Widget

4. In the expression editor that appears type a list of the PizzaToppings you would like to be considered vegetarian, separated by the union symbol (which can be selected off the palette, or added by typing “or” (lowercase).

5. Click the Protégé nerd face (or press return) to accept the changes. This can only be done when the face is not red.

6. Move the superclass, PizzaTopping, into the Necessary & Sufficient Conditions.

7. Classify your ontology – check that all the expected toppings are now subsumed by VegetarianTopping.
Exercise 6: Create VegetarianPizza – Universal Restrictions

To state that members of a class can only have a specific relationship with individuals from a specific class we use a Universal (AllValuesFrom) restriction.

We need to model the fact that:

Any Pizza that only has toppings from VegetarianTopping is a VegetarianPizza

1. Create a new subclass of Pizza called VegetarianPizza

2. Create an AllValuesFrom (Universal) restriction on this class along the hasTopping property with a filler of VegetarianTopping.

3. Convert this class into a defined class by dragging the conditions up to the Necessary & Sufficient heading.

4. Classify your ontology

Questions:
- What Happens? Do you get the expected results?
Exercise 7: Closing Pizzas
Because of the Open World Assumption we need to close our pizza descriptions in order for them to classify correctly under VegetarianPizza.

1. Select a primitive pizza that does not contain any meat topping (from subclasses of NamedPizza)

2. Create a new AllValuesFrom (Universal) restriction along the hasTopping property. The filler will be a union of all of the toppings on the pizza – i.e., the union should match all of the fillers of the existing SomeValuesFrom (Existential) restrictions using the hasTopping property. MargheritaPizza is given as an example below:

3. Classify your ontology to check if this class classifies under VegetarianPizza

4. Close all of your primitive pizzas. They should all contain a closure axiom (Universal restriction) similar to below. Classify each time to check that the closure has worked.
<table>
<thead>
<tr>
<th>Asserted Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NamedPizza</td>
</tr>
<tr>
<td>( \forall ) hasTopping (MozzarellaTopping \cup TomatoTopping)</td>
</tr>
<tr>
<td>B hasTopping MozzarellaTopping</td>
</tr>
<tr>
<td>B hasTopping TomatoTopping</td>
</tr>
</tbody>
</table>
Resources:

Including specs, pointers to other material, including applications and related projects

Protégé – [protege.stanford.edu](http://protege.stanford.edu)
See also the WIKI available at [http://protege.cim3.net/cgi-bin/wiki.pl](http://protege.cim3.net/cgi-bin/wiki.pl)
Especially useful for the mailing list – subscribe to Protégé-OWL for support and discussion

Protégé-OWL homepage - [http://protege.stanford.edu/plugins/owl/](http://protege.stanford.edu/plugins/owl/) includes most recent versions of the rapidly changing software, documentation, papers and details of the Java API as well as examples of producing plugins

CO-ODE – [www.co-ode.org](http://www.co-ode.org)
Plugins and other resources/materials available

Pizzas and other ontologies are available from: [www.co-ode.org/ontologies/](http://www.co-ode.org/ontologies/)

Semantic Web Best Practices Working Group
Producing documentation on ontology quality and other aspects such as identifying Design Patterns


WIKI - [http://esw.w3.org/topic/FrontPage](http://esw.w3.org/topic/FrontPage)