Protégé-OWL Tutorial

Session 2: Defined Classes

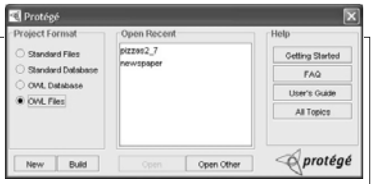
Nick Drummond

This session

- ► Issue: Primitive Classes & Polyhierarchies
- ► Advanced: Creating Defined Classes
- ► Reasoner: Classifying
- ► Union Classes: Covering Axioms
- ► Example: Creating a Vegetarian Pizza
- ► Issue: Open World Assumption
- ► Union Classes: Closure

Loading OWL files from scratch

Run Protégé.exe



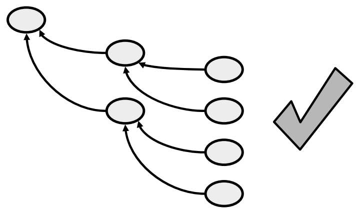
1. *If you've only got an OWL file:*
   Select "OWL Files" as the Project Format, then "Build" to select the .owl file
2. *If you've got a valid project file*:*
   Select "OWL Files" as the Project Format, and then "Open Other" to find the .pprj file (if you've already opened it, it will be in "Open Recent")
3. *Open C:\Protégé_3.0_beta\examples\pizzas\pizzas2_0.owl*

** 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*

Primitive Classes

- ► All classes in our ontology so far are Primitive
- ► We describe primitive pizzas
- ► Primitive Class = only Necessary Conditions
- ► They are marked as yellow in the class hierarchy

We condone building a disjoint tree of primitive classes



Describing Primitive Pizza Classes

Start with pizzas2_0.owl

1. *Create a new pizza under NamedPizza*
   either choose from the menu or make it up
2. *Create a new Existential (SomeValuesFrom) Restriction with the hasTopping property and a filler from* **PizzaTopping** *(eg* **HamTopping***)*
3. *Add more Restrictions in the same way to complete the description*
   each restriction is added to an intersection –
   so a Pizza must have toppingA ***and*** *must have toppingB etc*
   see **MargheritaPizza** *for an example*
4. *Create another pizza that has at least one meat ingredient*
   remember disjoints

Polyhierarchies

► By the end of this tutorial we intent to create a **VegetarianPizza**

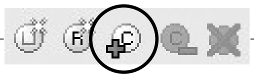
► Some of our existing Pizzas should be types of **VegetarianPizza**

► However, they could also be types of **SpicyPizza** or **CheeseLoversPizza**

► We need to be able to give them multiple parents

Vegetarian Pizza attempt 1

Start with pizzas2_1.owl

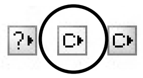


1. *Create a new pizza called "VegetarianPizza" under* **Pizza**
   make this disjoint from its siblings as we have been doing
2. *Select* **MargheritaPizza**
   you will notice that it only has a single parent, **NamedPizza**
3. *Add* **VegetarianPizza** *as a new parent using the conditions widget "Add Named Class" button*

   we have asserted that **MargheritaPizza** *has 2 parents*

Reasoning about our Pizzas

Start with your existing ontology

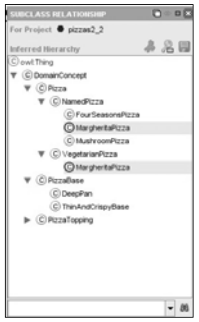


1. *Start RACER*
2. *Classify your ontology*
   You will see an inferred hierarchy appear, which will show any movement of classes in the hierarchy

   You will also see a results window appear at the bottom of the screen which describes the results of the reasoner

   MargheritaPizza *turns out to be inconsistent – why?*

   Remember **MeatyVegetableTopping**?



Attempting again

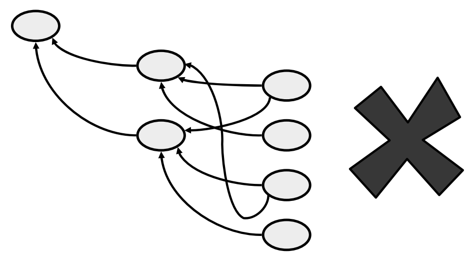
Close the inferred hierarchy and results

1. *Remove the disjoint between* **VegetarianPizza** *and its siblings*
   When prompted, choose to remove only between this class and its siblings
2. *Re-Classify your ontology*
   This should now be accepted by the reasoner with no inconsistencies

Asserted Polyhierarchies

► We believe asserting polyhierarchies is bad

► We lose some encapsulation of knowledge
► Difficult to maintain



let the reasoner do it!

Defined Classes

► Have a definition. That is *at least one* Necessary and Sufficient condition

► Are marked in orange in the interface

► Classes, all of whose individuals satisfy this definition, can be inferred to be subclasses

► Reasoners can perform this inference

Describing a MeatyPizza

Start with your existing ontology, *close the reasoner panes*

1. *Create a subclass of* **Pizza** *called* **MeatyPizza**
   Don't put in the disjoints or you'll get the same problems as before
   In general, defined classes are not disjoint from siblings
2. *Add a restriction to say:*
   "Every **MeatyPizza** *must have at least one meat topping"*
3. *Classify your ontology*
   What happens?

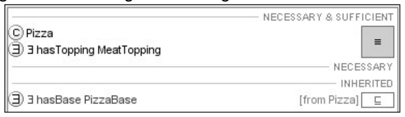
Defining a MeatyPizza

Start with your existing ontology, *close the reasoner panes*

1. *Click and drag your ∃ hasTopping* **MeatTopping** *restriction from "*Necessary*" to "*Necessary & Sufficient*"*
   The **MeatyPizza** *class now turns orange, denoting that it is now a defined class*
2. *Click and drag the* **Pizza** *Superclass from "*Necessary*" to "*Necessary & Sufficient*"*

   Make sure when you release you are on top of the existing restriction otherwise you will get 2 sets of conditions.

   You should have a single orange icon on the right stretching across both conditions like this…
3. *Classify your ontology*
   What happens?

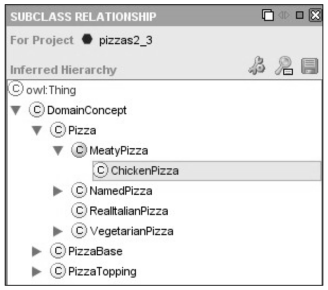

NECESSARY & SUFFICIENT
Pizza
∃ hasTopping MeatTopping ≡
NECESSARY
INHERITED
∃ hasBase PizzaBase [from Pizza] ⊑

Reasoner Classification

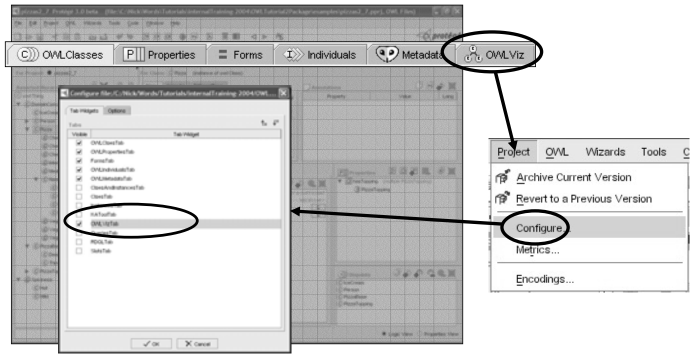
► The reasoner has been able to infer that anything that is a **Pizza** that has at least one topping from **MeatTopping** is a **MeatyPizza**

► Therefore, classes fitting this definition are found to be subclasses of **MeatyPizza**, or are subsumed by **MeatyPizza**

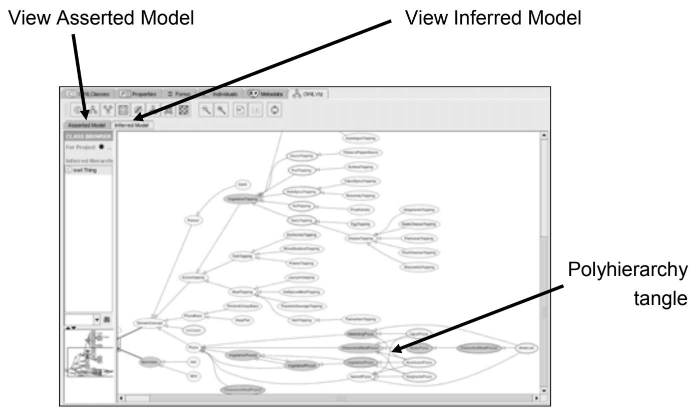
► The inferred hierarchy is updated to reflect this and moved classes are highlighted in blue


SUBCLASS RELATIONSHIP
For Project ● pizzas2_3
Inferred Hierarchy
- owl:Thing
  - DomainConcept
    - Pizza
      - MeatyPizza
        - ChickenPizza
      - NamedPizza
      - RealItalianPizza
      - VegetarianPizza
    - PizzaBase
    - PizzaTopping

Viewing our Hierarchy Graphically



OWLViz Tab



How do we Define a Vegetarian Pizza?

- ► Nasty
- ► Define in words?
  - ► “a pizza with only vegetarian toppings”?
  - ► “a pizza with no meat (or fish) toppings”?
  - ► “a pizza that is not a MeatyPizza”?
- ► More than one way to model this

Defining a Vegetarian Topping

Start with your existing ontology

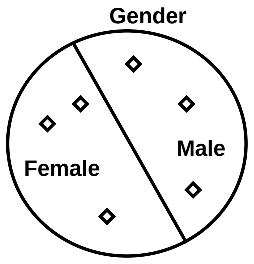
1. *Create a subclass of* **PizzaTopping** *called* **VegetarianTopping**
2. *Click “Create New Expression” in the Conditions Widget*
   Type in or select each of the top level **PizzaTopping***s that are not meat or fish (ie* **DairyTopping**, **FruitTopping** *etc) and between each, type the word “or”*
   the “or” will be translated into a union symbol
3. *Press Return when finished*
   you have created an anonymous class described by the expression
4. *Make this a defined class by moving both conditions from the “Necessary” to the “Necessary & Sufficient” conditions*
5. *Classify your ontology*

Class Constructors: Union

- ► AKA “disjunction”
- ► This OR That OR TheOther
- ► (This⊔That⊔TheOther)
- ► Set theory
- ► Commonly used for:
  - ► Covering axioms (like **VegetarianTopping**)
  - ► Closure

Covering Axioms

- ► Covered class – that to which the condition is added
- ► Covering classes – those in the union expression
- ► A covering axiom in the “Necessary & Sufficient” Conditions means:
  the covered class cannot contain any instances from a class other than one of the covering classes



Gender ≡ Female⊔Male

In this example, the class Gender is “covered” by Male or Female

All individuals in Gender must be individuals from Male or Female

There are no other types of Gender

Vegetarian Pizza attempt 2

Start with your existing ontology

1. *Select* **MargheritaPizza** *and remove* **VegetarianPizza** *from its superclasses*
2. *Select* **VegetarianPizza** *and create a restriction to say that it "only has toppings from* **VegetarianTopping***"*
3. *Make this a defined class by moving all conditions from "*Necessary*" to "*Necessary & Sufficient*"*
   Make sure when you release you are on top of the existing restriction otherwise you will get 2 sets of conditions.
   You should have a single orange icon on the right stretching across both conditions
4. *Classify your ontology*
   What happens?

Open World Assumption

- The reasoner does not have enough information to classify pizzas under **VegetarianPizza**
- Typically several Existential restrictions on a single property with different fillers – like primitive pizzas
- Existential should be paraphrased by "amongst other things…"
- Must state that a description is complete
- We need closure for the given property
- This is in the form of a Universal Restriction with a Union of the other fillers using that property

Closure

- Example: **MargheritaPizza**

  All **MargheritaPizzas** must have:

  at least 1 topping from **MozzarellaTopping** and
  at least 1 topping from **TomatoTopping** and
  only toppings from **MozzarellaTopping** or **TomatoTopping**
- The last part is paraphrased into

  "no other toppings"
- The union closes the hasTopping property on **MargheritaPizza**

Closing Pizza Descriptions

Start with your existing ontology

1. *Select* **MargheritaPizza**
2. *Create a Universal Restriction on the hasTopping property with a filler of "***TomatoTopping ⊔ MozzarellaTopping***"*
   Remember, you can type "or" to achieve this, or you can use the expression palette
3. *Close your other pizzas*
   Each time you need to create a filler with the union of all the classes used on the hasTopping property (ie all the toppings used on that pizza)
4. *Classify your ontology*
   Finally, the defined class **VegetarianPizza** *should subsume any classes that only have vegetarian toppings*

Other Definitions of Veggie Pizzas

Start with your existing ontology

1. *Create a VegetarianPizza2*
2. *Create a new class expression ”not MeatyPizza”*
   You can type “not” to achieve this, or you can use the expression palette
3. *Convert this class to a defined class*
   Move both Pizza and NOT MeatyPizza to the necessary & sufficient conditions
4. *Classify your ontology*
   Finally, the defined class **VegetarianPizza2** *should subsume any classes that are not MeatyPizzas*

Other Definitions of Veggie Pizzas

Start with your existing ontology

1. *Create a VegetarianPizza3*
2. *Create a restriction to state that it has some MeatTopping*
3. *Negate this by editing the class expression (double click on it)*
4. *Convert this class to a defined class*
5. *Classify your ontology*

Other Definitions of Veggie Pizzas

Start with your existing ontology

1. *Create a VegetarianPizza4*
2. *Create a universal restriction on the hasTopping property with a a filler of “NOT MeatTopping”*
3. *Convert this class to a defined class*
4. *Classify your ontology*
5. *Note that this is an equivalent class to the previous definition of VegetarianPizza*
   – “∀ hasTopping NOT meat”
   – “NOT ∃ hasTopping meat”

Properties: Domain and Range

Start with your existing ontology

1. *Add a domain of Pizza to hasTopping*
2. *Create a class Icecream under DomainConcept*
3. *Create a restriction to state that Icecream has at least one topping from CheeseTopping (yuk!)*
4. *Classify your ontology*
5. *What has happened to Icecream?*
6. *Make sure Icecream is disjoint to its siblings and reclassify*

Properties: Domain and Range

- ► Domain and Range are not used to restrict the interface
- ► They are used by the reasoner to infer additional information about individuals
- ► An individual that uses a property with a domain set can be inferred to be a member of the domain class (the same holds for range)
- ► Classes that uses a property with a domain set in an existential restriction will be inferred to be a subclass of the domain class (or inconsistant if these classes are disjoint)
  - the same does not apply to range.
  This is because all individuals in this class must have at least one relationship using this property

Properties: Functional Properties

Start with your existing ontology

1. *Make hasTopping functional*
2. *Classify your ontology – what happens?*
3. *Undo your change to hasTopping, but make hasBase functional*

Other Exercises: Create an InterestingPizza

1. *Create a class, InterestingPizza under Pizza*
2. *Create a minCardinality restriction for hasTopping with a value of 3*
3. *Convert to a defined class*
4. *Classify to check that it works – you may need to add some more pizzas*

Other Exercises: Define RealItalianPizza

- *Convert* **RealItalianPizza** *to a defined class*
- *Add information to your pizzas to allow some of them to classify under this one*
- *Classify*
  remember to check your disjoint if you have problems

Other Exercises

- ► Add FishTopping to PizzaToppings, and correct your vegetarian pizzas to use “MeatTopping OR FishTopping”
- ► Create other PizzaToppings
- ► Create other Pizzas from the menu and check that they classify correctly

Summary

You should now be able to:

- ► Use Defined Classes to allow a polyhierarchy to be computed
- ► Classify using a Reasoner
- ► Create Covering Axioms
- ► Close Class Descriptions to cope with Open World Reasoning
- ► Appreciate that there can be several ways to model a class (some of which are exactly equivalent)
- ► Understand the effects of using domain and range on properties

Your Pizza Finder

- ► Once you have a pizza ontology you are happy with, you can “plug it in” to the PizzaFinder
- ► Instructions available on line at…

The Software

- ►Protégé is Free
- ►The OWL Plugin is Open Source
- ►Many plugins are available from scripting to visualisation
- ►Software / resources / community at:
  - ►http://protege.stanford.edu/
  - ►http://www.co-ode.org/