

# OWL Pizzas:

## Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns

Alan Rector<sup>1</sup>, Nick Drummond<sup>1</sup>, Matthew Horridge<sup>1</sup>, Jeremy Rogers<sup>1</sup>, Holger Knublauch<sup>2</sup>, Robert Stevens<sup>1</sup>, Hai Wang<sup>1</sup>,  
Chris Wroe<sup>1</sup>

<sup>1</sup>Information Management Group / Bio Health Informatics Forum  
Department of Computer Science, University of Manchester

<sup>2</sup>Stanford Medical Informatics, Stanford University

[rector@cs.man.ac.uk](mailto:rector@cs.man.ac.uk)

[co-ode-admin@cs.man.ac.uk](mailto:co-ode-admin@cs.man.ac.uk)

[www.co-ode.org](http://www.co-ode.org)  
[protege.stanford.org](http://protege.stanford.org)

# Why do so few people use OWL and DLs?

## Why so little use of classifiers?

*Is part of the answer that...*

- OWL/DLs run counter to common intuitions from
  - Databases, UML, query languages (including RDQL)
  - Logic programming & rule systems, e.g. JESS, PAL
  - Frame systems – more difference than at first appears
  - Object oriented programming
- Can Tools can help?
  - Can we use tutorials and training to gather requirement?
    - All examples here have occurred repeatedly in practice in tutorials or in live ontology construction – often by experts in other formalisms
      - Part of the requirements gathering for the Protégé-OWL interface

# OWL Pizzas Tutorial

- **Designed to address common errors**
  - **We have seen lots of experienced people make the same simple mistakes**
- **Why Pizzas?**
  - **Naturally combinatorial**
  - **No serious ontological issues**
  - **Familiar and fun (at least to western audiences)**
  - **Easy to illustrate most problems**
- **Extended version**
  - **See 120 pg ‘textbook’ version on <http://www.co-ode.org>**

# Issues and common errors

- Open world reasoning
  - Domain and range constraints as axioms
  - Trivial satisfiability of universal restrictions
  - Subsumption (“is kind of”) as necessary implication
- Unfamiliar constructs – confusing notation/terminology
  - Confusion of universal (*allValuesFrom*) rather than existential restrictions (*someValuesFrom*)
  - Need for explicit disjointness axioms
- Errors in understanding common logical constructs
  - Confusing ‘and’ and ‘or’
  - Defined vs primitive classes & conversion between them
  - Use of subclass axioms as rules
- Understanding the effect of classification
  - What to do when it all turns red – debugging
  - Explaining classification

# Open World Reasoning

## “Vegetarian Pizzas”

*The menu says that:*

- “Margherita pizzas have tomato and mozzarella toppings”
- “Vegetarian pizzas have no meat or fish toppings”

*What's it mean?*

# Three Views from Protégé OWL tools

The image shows a screenshot of the Protégé OWL editor interface, displaying three different views of the 'Magherita\_Pizza' class.

**Left Panel (RDFS:COMMENT):**

Magherita\_Pizza

RDFS:COMMENT:

A magherita pizza kind of pizza and has has, amongst other things, Mozzarella and Tomato toppings

Asserted Inferred

ASSERTED CONDITIONS:

NECESSARY

- ⊂ Pizza
- ⊃ ∃ has\_topping Mozzarella\_topping
- ⊃ ∃ has\_topping Tomato\_topping

**Top Right Panel (Paraphrase/descriptive syntax):**

Paraphrase/descriptive syntax

Class: Magherita\_Pizza

NECESSARILY

Pizza

has\_topping some Mozzarella\_topping

has\_topping some Tomato\_topping

**Bottom Right Panel (OWL Abstract Syntax):**

OWL Abstract Syntax

```
Class(Magherita_Pizza partial
  Pizza
  restriction(has_topping someValuesFrom(
    Mozzarella_topping))
  restriction(has_topping someValuesFrom(
    Tomato_topping)))
```

# Vegetarian Pizza

Name

Vegetarian\_pizza

rdfs:comment

A vegetarian pizza is ANY pizza that has no meat topping and no fish topping.

Asserted

Inferred

Asserted Conditions

NECESSARY &

C

Pizza

¬


∃

has\_topping Fish\_topping

¬

∃

has\_topping Meat\_topping

 Class Description

Paraphrase/descriptive syntax ▼

Class: Vegetarian\_pizza

DEFINITION

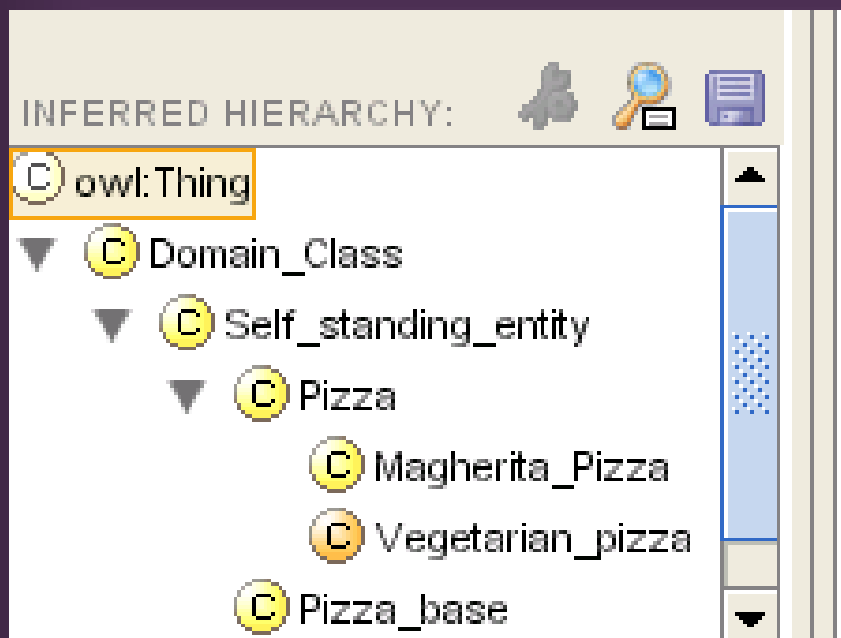
Pizza

NOT(has\_topping some Meat\_topping)

NOT(has\_topping some Fish\_topping)

# Is a Margherita Pizza a Vegetarian Pizza?

- Not according to classifier



- And not according to the full paraphrases formulated carefully



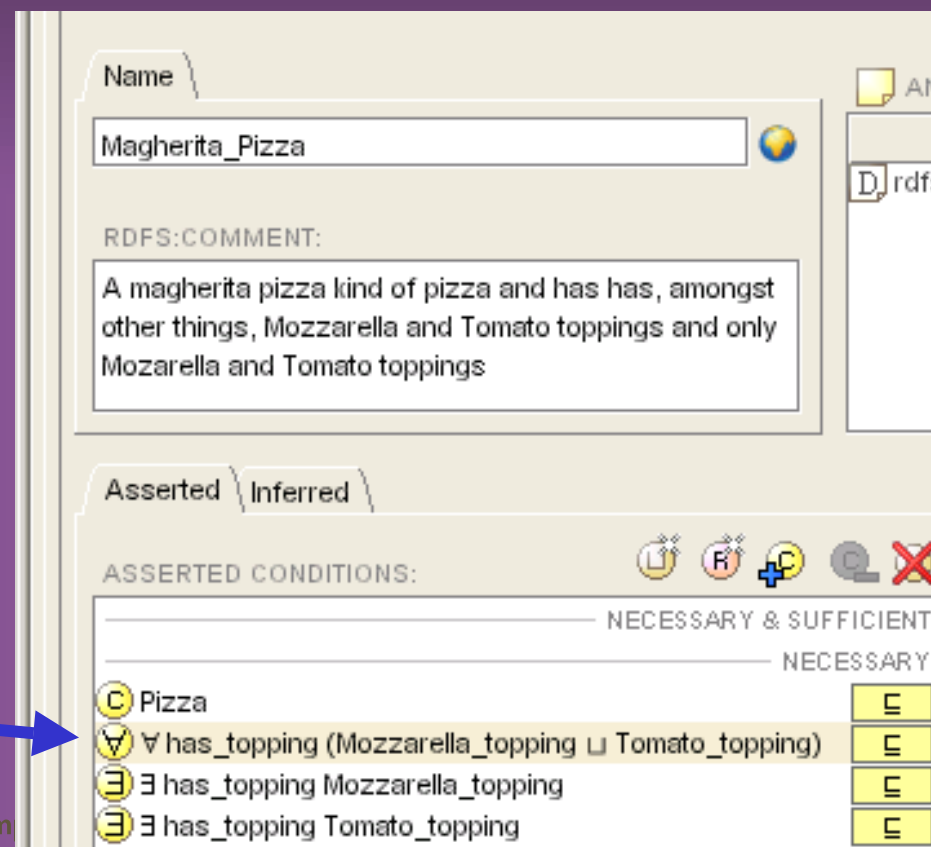
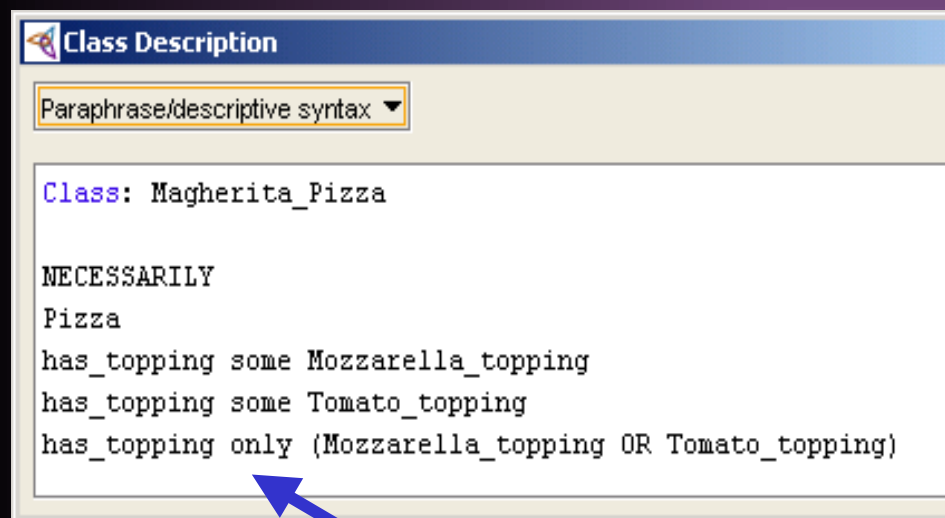
# Open World Reasoning

## Vegetarian & Margherita Pizzas

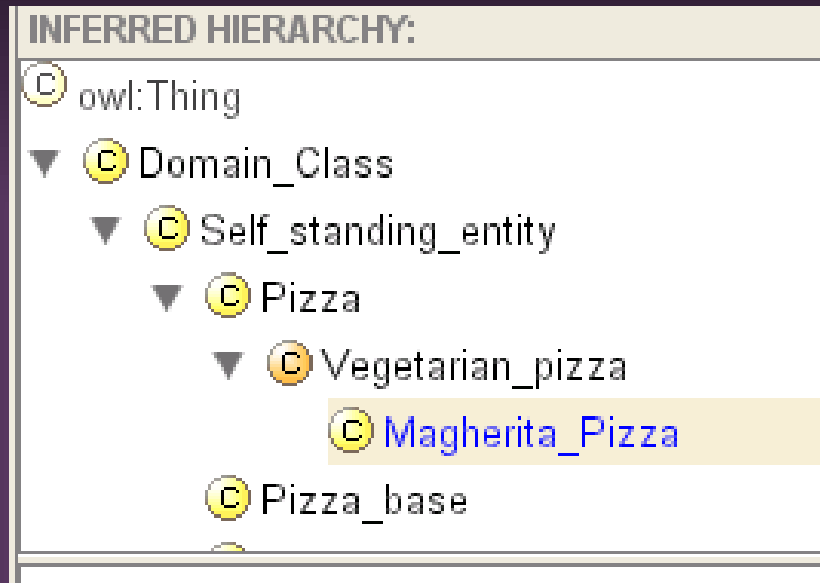
- “A vegetarian pizza is *any* pizza that, *amongst other things*,  
*does not* have *any* meat topping *and*  
*does not* have *any* fish topping”
- “A margherita pizza is *a* pizza and, *amongst other things*,  
has *some* tomato topping *and*  
has *some* mozzarella topping”

# Add “Closure Axiom”

- “A Margherita pizza has tomato and cheese toppings and only tomato and cheese toppings”
  - i.e. “A Margherita pizza has tomato and cheese toppings and only toppings that are tomato *or* cheese”
  - Tedious to create by hand, so provide automatic generation in tool



# Now Classifies as Intended



- **Provided:**  
Toppings mutually disjoint

# Domain & Range Constraints

- Actually axioms
  - *Property P range( RangeClass )*  
means
    - owl:Thing  
restriction(P *allValuesFrom* RangeClass)
  - *Property P domain( DomainClass )*  
means
    - owl:Thing  
restriction(inverse(P) *allValuesFrom* DomainClass)

# Non-Obvious Consequences

- Range constraint violations – *unsatisfiable or ignored*
  - If filler and RangeClass are disjoint: *unsatisfiable*
  - Otherwise nothing happens!
- Domain constraint violations – *unsatisfiable or coerced*
  - If subject and DomainClass are disjoint: *unsatisfiable*
  - Otherwise, subject reclassified (coerced) to kind of DomainClass!
- Furthermore cannot be fully checked before classification
  - although tools can issue warnings.

# Example of Coercion by Domain violation

- has\_topping: *domain(Pizza) range(Pizza\_topping)*

class Ice\_cream\_cone  
has\_topping some Ice\_cream

- If Ice\_cream\_cone and Pizza are *not* disjoint:
  - Ice\_cream\_cone is classified as a kind of Pizza
    - ...but: Ice\_cream is *not* classified as a kind of Pizza\_topping
  - Have shown that:
    - all* Ice\_cream\_cones are a kinds of Pizzas,
    - but only that:
      - some* Ice\_cream is a kind of Pizza\_topping

» *Only domain constraints can cause reclassification*  
... by now most people are very confused - need lots of examples & back to basics

# Subsumption means necessary implication

- “*B is a kind of A*”  
means  
“*All Bs are As*”
  - “*Ice\_cream\_cone is a kind of Pizza*”  
means  
“*All ice\_cream\_cones are pizzas*”
    - From “Some Bs are As” we can deduce very little of interest in DL terms
      - » “some ice\_creams are pizza\_toppings”  
says nothing about “all ice creams”

# Trivial Satisfiability: More unintuitive results

- An existential (someValuesFrom) restriction with an empty filler makes no sense:
  - is unsatisfiable if its filler is unsatisfiable
- A Universal (allValuesFrom) restriction with an unsatisfiable filler is trivially satisfiable
  - provided there is no way to infer a existence of a filler
    - Leads to errors being missed and then appearing later

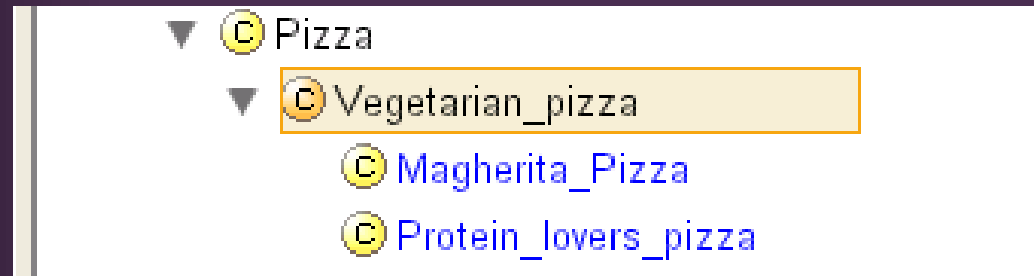


# Examples of Trivial Satisfaction

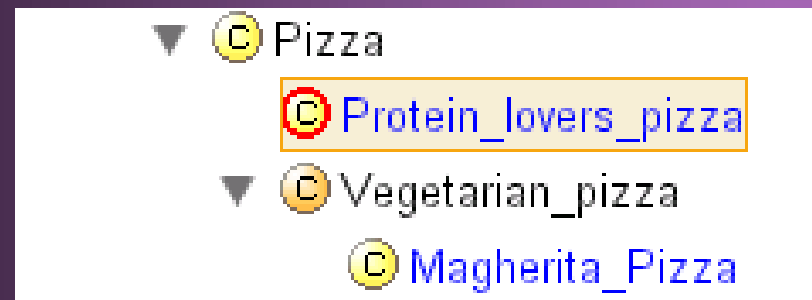
- **Unsatisfiable filler:**  
disjoint(Meat\_topping Fish\_topping)  
class(Protein\_lovers\_pizza complete  
has\_topping *allValuesfrom* (Meat\_topping *and* Fish\_topping))
  - i.e. *intersectionOf*(Meat\_topping, Fish\_topping)
  - i.e. *only something that is both* (Meat\_topping *and* fish\_topping)
- **Range constraint violation:**  
disjoint(Ice\_cream, Pizza\_topping)  
class(Ice\_cream\_pizza  
has\_topping *allValuesFrom* Ice\_cream)
- **Both legal unless/until there is an axiom such as:**  
Pizza has\_topping *someValuesFrom* Pizza\_topping
  - i.e. “All pizzas have at least one topping”

# Worse, Trivially Satisfied Restrictions Classify under Anything

- Protein\_lovers\_pizza *is a kind of* Vegetarian\_Pizza!



- Until we add:  
Pizza has\_topping some Pizza\_topping
  - “All pizzas have some topping”




*“Only  
does not  
imply  
some!”*

# The trouble with confusing “some” with “only”

## *someValuesFrom* with *allValuesFrom*

- It works for a while
  - The student defining **Protein\_lovers\_pizza** thought they were defining a pizza with meat toppings and fish toppings
- Errors only show up later when existentials are added elsewhere

**CLASS EDITOR**

FOR CLASS:  Protein\_lovers\_pizza (




**RDFS:COMMENT:**

A Pizza consisting of meat and fish toppings

Asserted Inferred

**INFERRED CONDITIONS:**

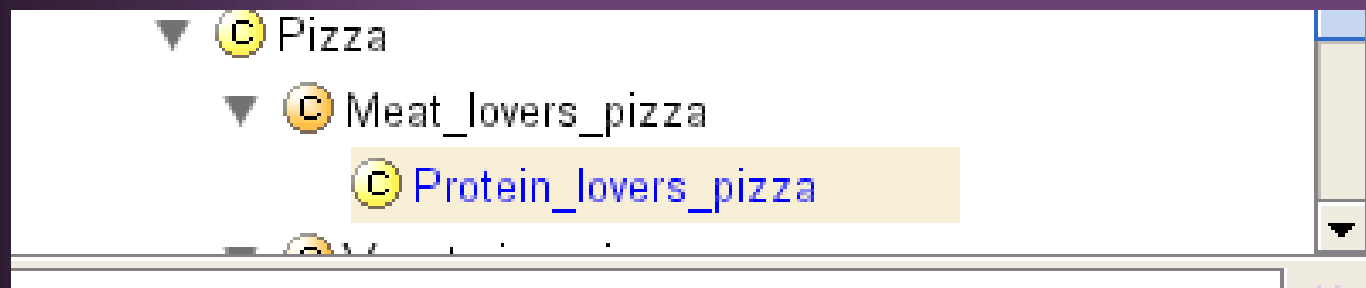
NECESSAR

-  Pizza
-   $\forall$  has\_topping Meat\_topping
-   $\forall$  has\_topping Fish\_topping

# The trouble with confusing “some” with “only”

## *someValuesFrom* with *allValuesFrom*

- Even classification seems to work at first
  - `class(Meat_lovers_pizza complete has_topping only Meat_topping )`



- So people continue complacently
  - Until the unexpected happens, e.g.
    - It is also classified as a kind of vegetarian pizza
    - It is made unsatisfiable by an existential axiom someplace

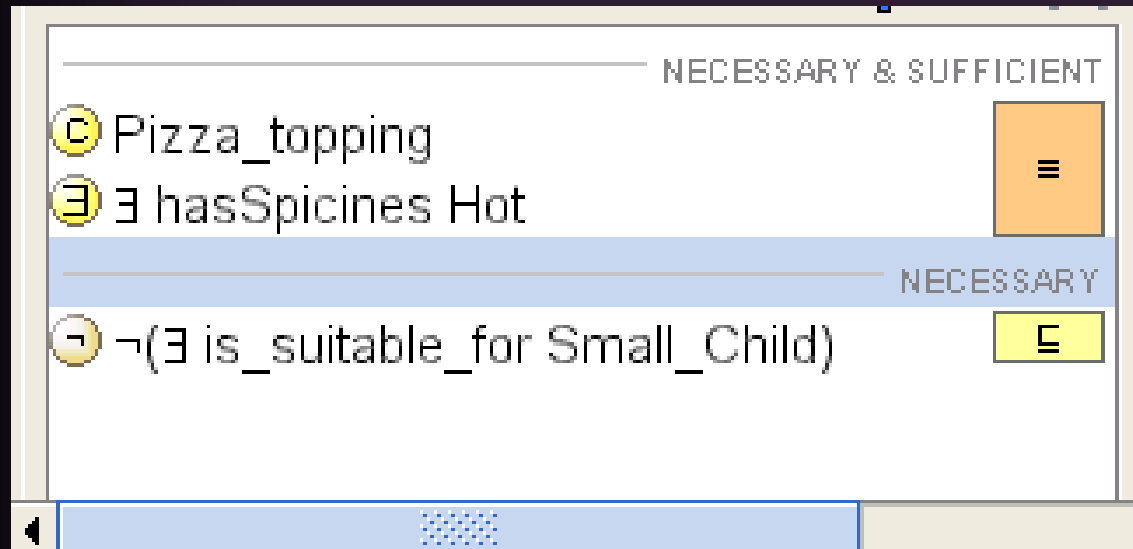
# Defined vs Primitive Classes

- In OWL the difference is a single keyword
  - “partial” vs “complete”
- In OilEd it was a single button
  - “subclass” vs “same class as” or “partial” vs “complete”
- Also...

Any necessary restrictions on defined classes must appear in separate subclassOf axioms

  - Breaks the object oriented paradigm
    - Hides information about the class on a different pane
  - Makes migrating a primitive class to a defined class tedious
    - Unless all restrictions become part of the definition
  - Makes subclass axioms for implication hard to understand

# Protégé-OWL – Everything in one place



**Necessary & Sufficient conditions:**

**“Definition”**

**Necessary conditions:**

**“Description”**

- **Spicy\_Pizza\_topping**  
Necessary & Sufficient:  
Pizza\_topping &  
has\_spiciness *some* Hot  
Necessarily also  
*Not* suitable\_for *any* Small\_child

# Defined classes

- Have necessary and sufficient conditions

# Primitive classes

- Have only necessary conditions
  - The necessary and sufficient space is empty

# Defined

At least one  
Necessary & Sufficient  
condition

NECESSARY & SUFFICIENT

- ☒ Pizza
- ☐  $\neg(\exists \text{ has\_topping Fish\_topping})$
- ☐  $\neg(\exists \text{ has\_topping Meat\_topping})$

NECESSARY

# Primitive

No  
Necessary & Sufficient  
conditions

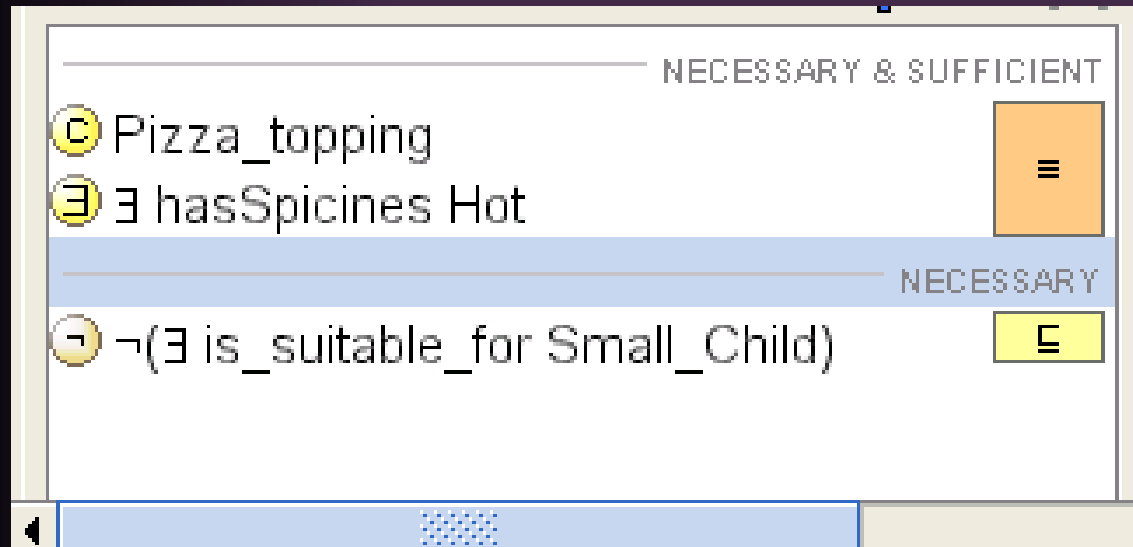
NECESSARY & SUFFICIENT

NECESSARY

- ☒ Pizza
- ☒  $\forall \text{ has\_topping (Mozzarella\_topping } \sqcup \text{ Tomato\_topping)}$
- ☐  $\exists \text{ has\_topping Mozzarella\_topping}$
- ☐  $\exists \text{ has\_topping Tomato\_topping}$



# Defined classes with necessary conditions



Necessary & Sufficient conditions:

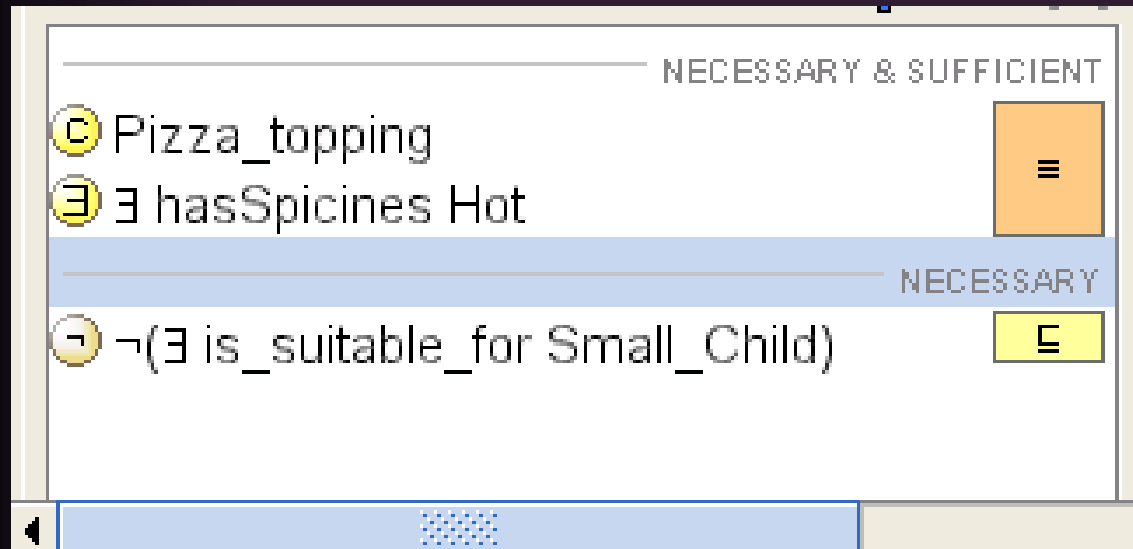
“Definition”

Necessary conditions:

“Description”

- In effect this is a rule
  - IF **Pizza\_topping** and **hasSpiciness** *some* Hot  
THEN **not suitable\_for** *any* small\_child
    - Easier to understand than separate subclass axioms.

# Protégé-OWL – Moving Conditions



**Necessary & Sufficient conditions:**

**“Definition”**

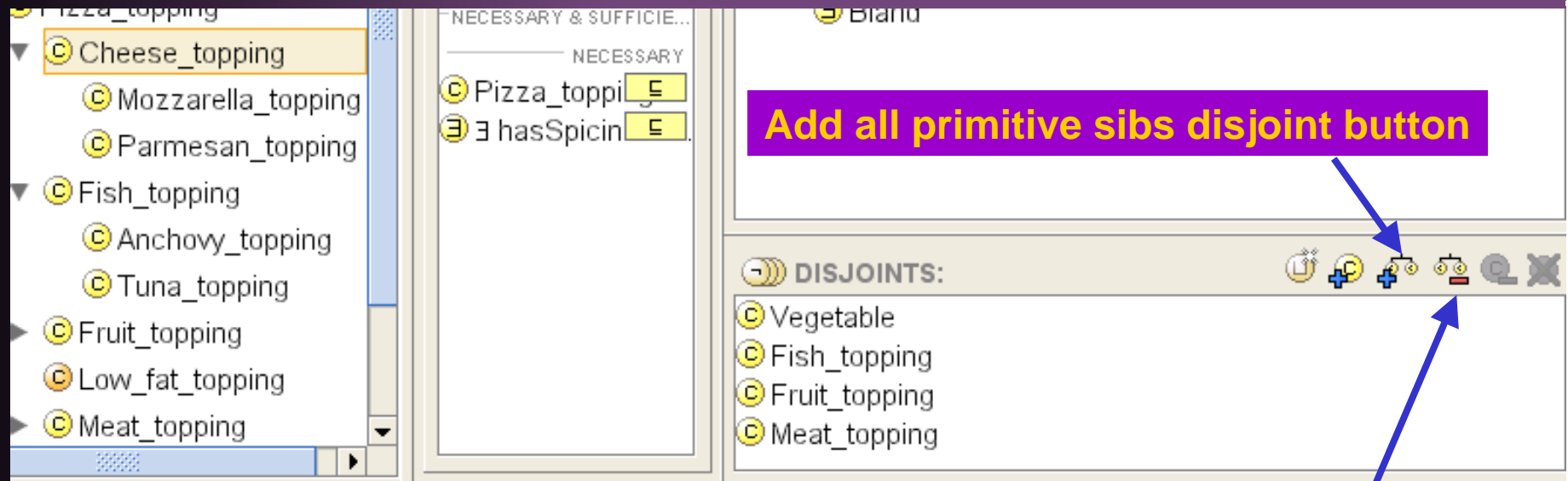
**Necessary conditions:**

**“Description”**

- **A common operation so:**
  - Cut & Paste
  - Drag and Drop
  - One click – convert to/from defined/primitive class

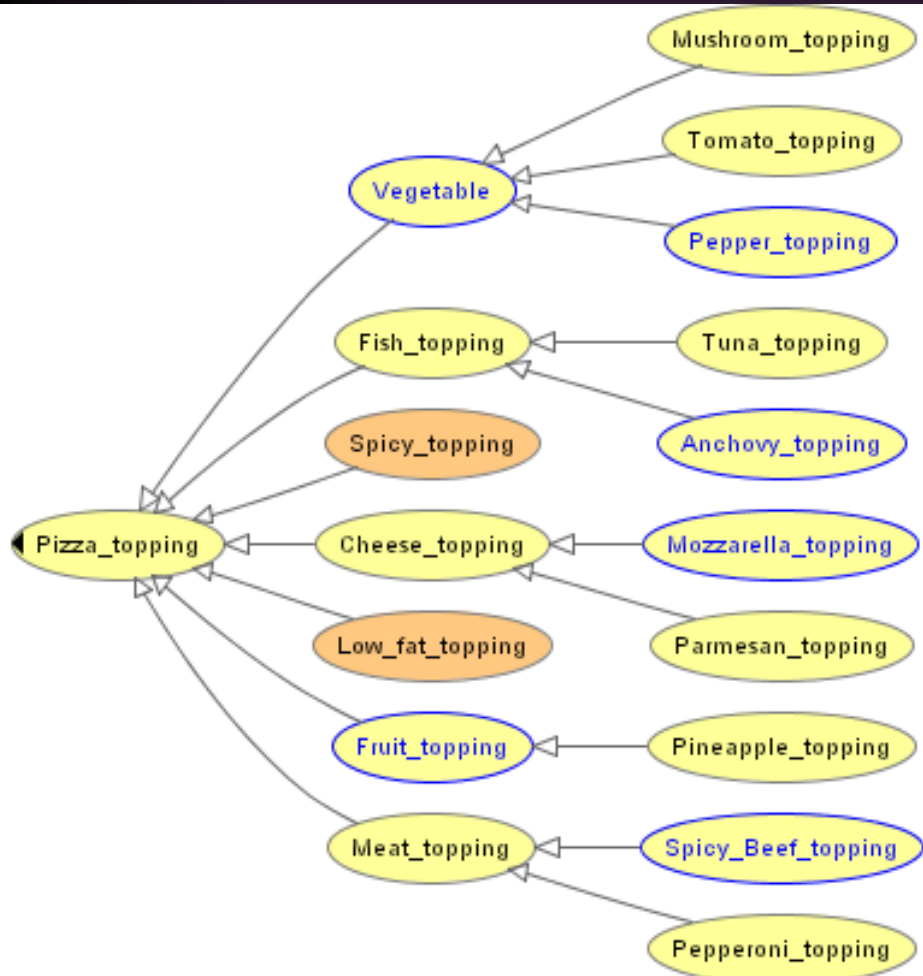
# Managing Disjointness

- **Basic; Must be explicit; Easy to forget**  
**So make it easy to do**
  - **Disjoint primitive siblings button**
  - **“Create group of classes” Wizard**
  - **Annotate parent – all primitive children disjoint**



**Remove all primitive sibs disjoint button**

# Understanding Classification



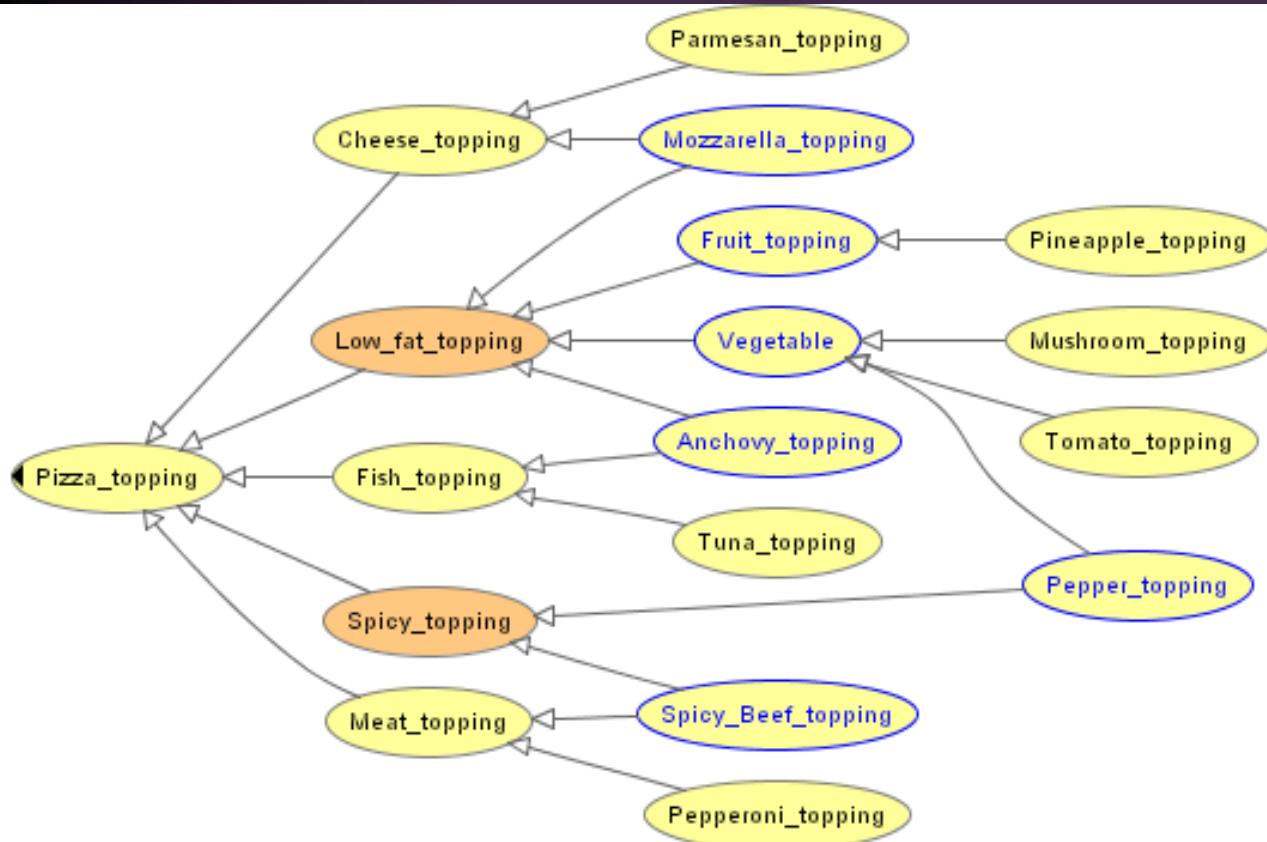
- Asserted
  - Simple tree
    - Defined (orange) classes have no children

# Understanding classification

- Inferred

- Polyhierarchy

- Defined (orange) classes have children

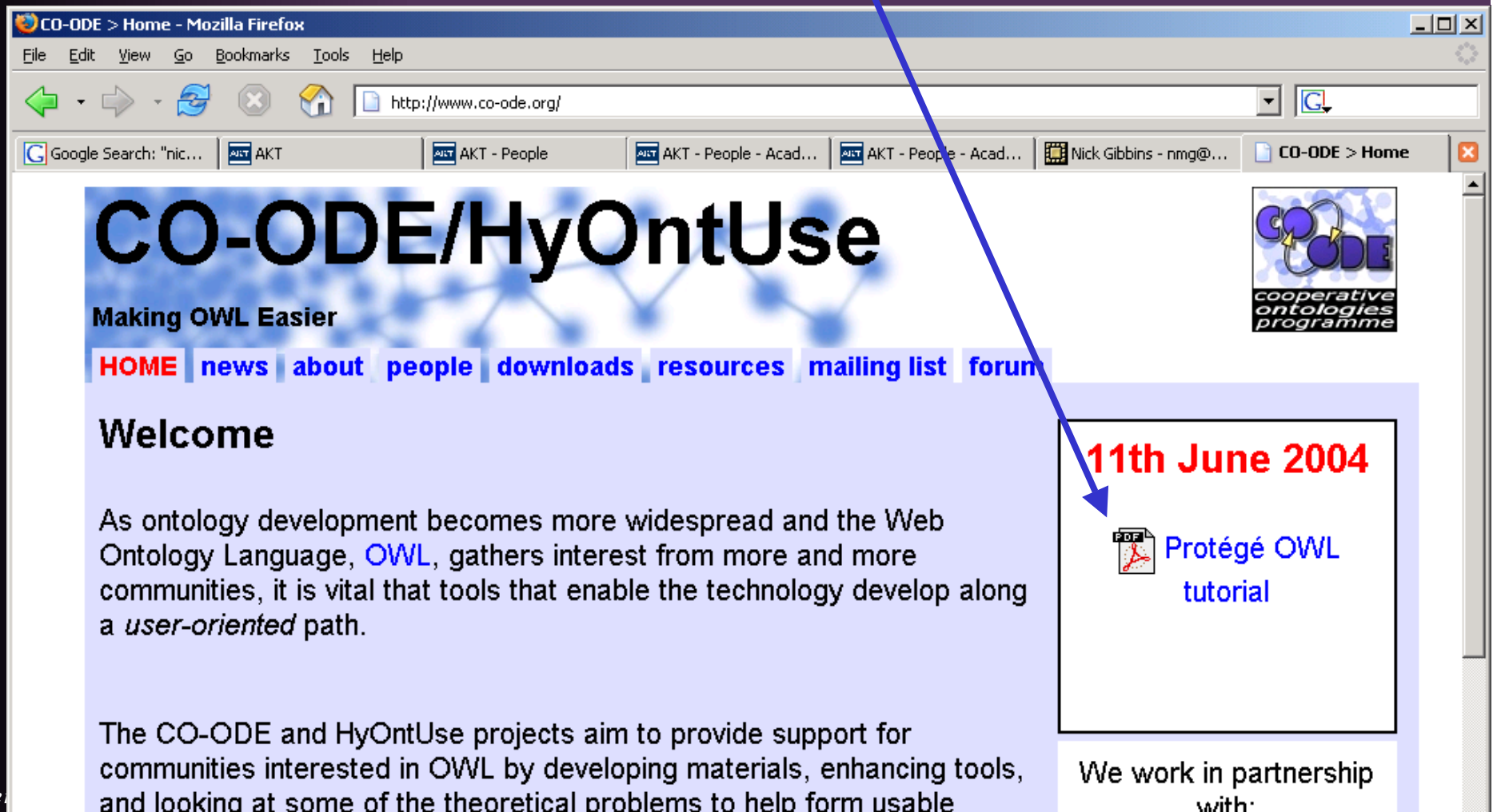


# What to do when “Its all turned red”

***Don't Panic!***

- Unsatisfiability propagates – so trace it to its source
  - Any class with an unsatisfiable filler in a *someValuesFor* condition is unsatisfiable
  - Any *subclass* of an unsatisfiable class is unsatisfiable
- Only a few possible sources
  - Violation of disjoint axioms
  - Unsatisfiable expressions
    - Confusion of “and” and “or”
  - Violation of a universal (*allValuesFrom*) constraint (including range and domain constraints)
    - Unsatisfiable domain or range constraints
- Tools coming RSN

# Web Site version 120 pp “Text book style” [www.co-ode.org](http://www.co-ode.org)



CO-ODE > Home - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.co-ode.org/

Google Search: "nic... AKT AKT - People AKT - People - Acad... AKT - People - Acad... Nick Gibbins - nmg@... CO-ODE > Home

# CO-ODE/HyOntUse

Making OWL Easier


[HOME](#) [news](#) [about](#) [people](#) [downloads](#) [resources](#) [mailing list](#) [forum](#)

## Welcome

As ontology development becomes more widespread and the Web Ontology Language, [OWL](#), gathers interest from more and more communities, it is vital that tools that enable the technology develop along a *user-oriented* path.

The CO-ODE and HyOntUse projects aim to provide support for communities interested in OWL by developing materials, enhancing tools, and looking at some of the theoretical problems to help form usable

**11th June 2004**

 [Protégé OWL tutorial](#)

We work in partnership  
with:





# *What's it Mean?*

- Paraphrases help clarify meaning
  - **someValuesFrom** “some”
  - **allValuesFrom** “only”
  - **complete** “A ... is any ... that...”
  - **partial** “All ... are...have...”
  - **negation** “does not have ... any...”
  - **intersection** “and” / “and also”
  - **union** “or” / “and/or”
  - **not...someValuesFrom** “not...any”
  - **not...allValuesfrom** “does not ...have only...”
  - **open world** “amongst other things”