

Master M2 IAFA 2022-2023

UE « Représentation des connaissances : Logique modale et ontologies » KINIAAE1

Enseignants : Emiliano Lorini, Andreas Herzig, Adrien Barton

Cours AB.1

Le Web Sémantique et ses techniques

Introduction aux ontologies

Adrien Barton

adrien.barton@gmail.com

CNRS, IRIT

13 Janvier 2023

Protégé : un outil de création d'ontologies

The screenshot displays the Protégé ontology editor interface. The top navigation bar includes tabs for 'Active Ontology', 'Entities', 'Classes', 'Object Properties', 'Data Properties', 'Annotation Properties', 'Individuals', 'OWLviz', 'DL Query', 'OntoGraf', 'Ontology Differences', and 'SPARQL Query'. The 'Entities' tab is active, and the 'Class hierarchy' sub-tab is selected.

The main left pane shows the 'Class hierarchy: 'drug administration specification'' tree. The hierarchy is as follows:

- 'information content entity'
 - 'condition'
 - 'data item'
 - 'datum label'
 - 'directive information entity'
 - 'dose specification'
 - 'normative specification'
 - 'drug administration specification' (selected)
 - 'drug dispensing specification'
 - 'prescribed dosing specification'
 - 'plan specification'
 - 'selection criterion'
 - 'dose quantification specification'
 - 'dose range specification'
 - 'drug dispensing amount specification'
 - 'drug product specification'
 - 'duration of administration specification'
 - 'prescribed drug product characteristic specification'
 - 'rate of administration specification'
 - 'route of administration specification'
 - 'site of administration specification'

The right pane shows the 'Annotations: 'drug administration specification'' section. It includes:

- label** [language: en]: drug administration specification
- definition** [language: en]: A normative specification that specifies how to perform a drug administration. It specifies:
 - The drug product
 - The posology
 - The condition(s) for starting
- definition** [language: fr]: Une entité informationnelle directive indiquant l'administration d'un médicament. Elle indique :

Below this, the 'Description: 'drug administration specification'' section shows:

- Equivalent To**: +
- SubClass Of**: +
 - 'has part' **some** 'drug product specification'
 - 'has part' **some** 'prescribed dosing specification'
 - 'has part' **some** 'starting drug administration condition'
 - 'normative specification'
- SubClass Of (Anonymous Ancestor)**:
 - 'is about' **some** 'realizable entity'
 - 'is about' **some** entity
- Members**: +
- Target for Key**: +

At the bottom of the interface, a status bar indicates: 'To use the reasoner click Reasoner->Start reasoner' and a checked checkbox for 'Show Inferences'.

Tutoriel

Adapté de “A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools, Edition 1.3”, de Matthew Horridge

http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf

Individus et propriétés



Figure 3.1: Representation Of Individuals

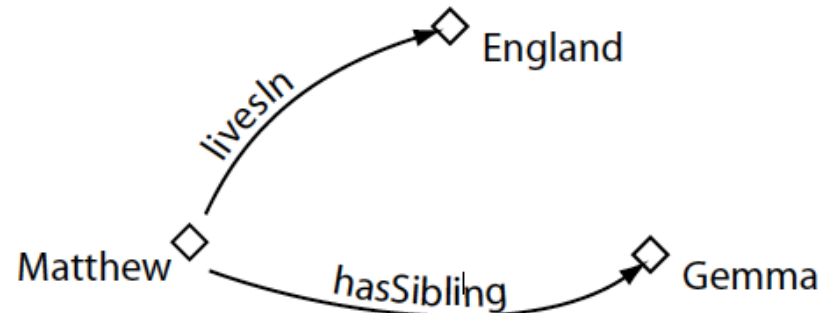


Figure 3.2: Representation Of Properties

Classes

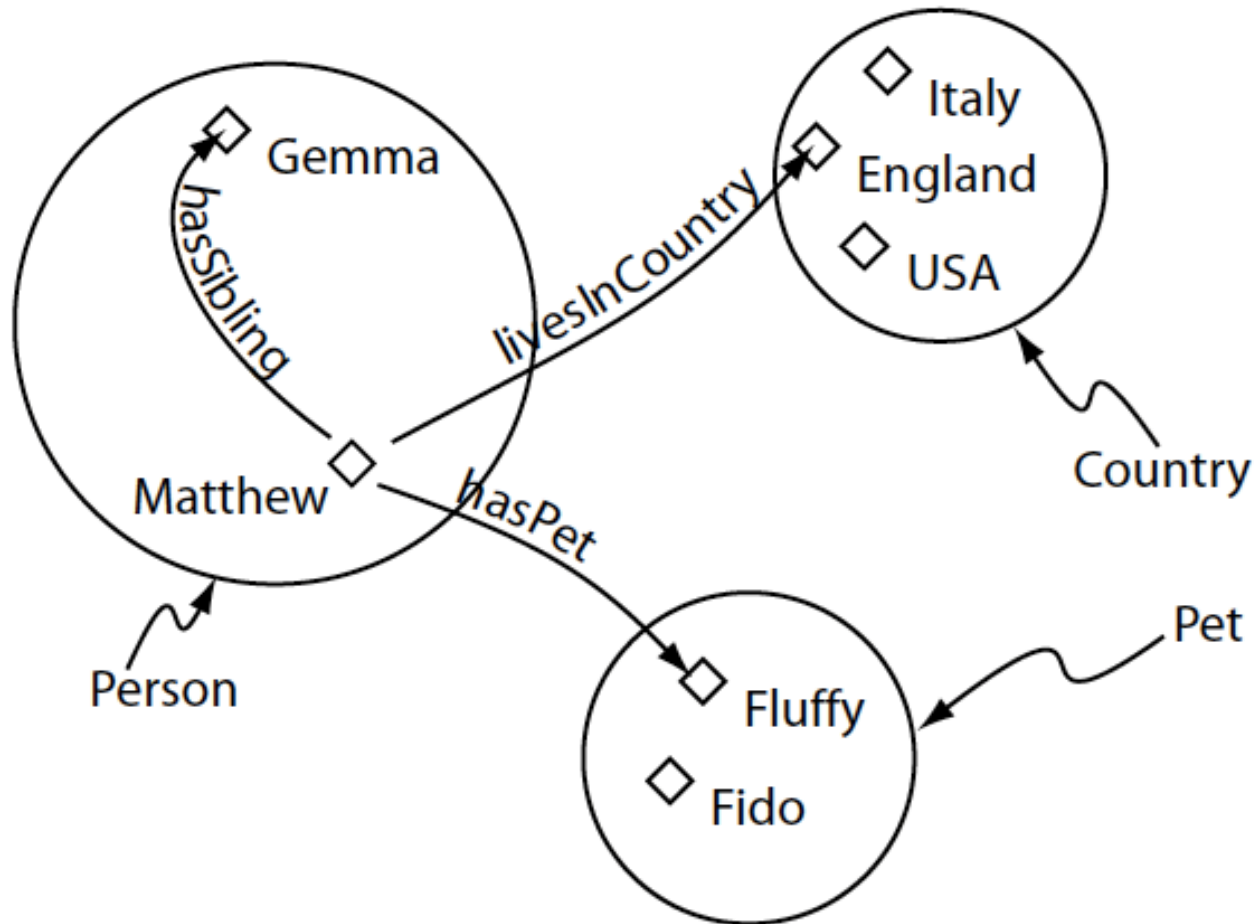
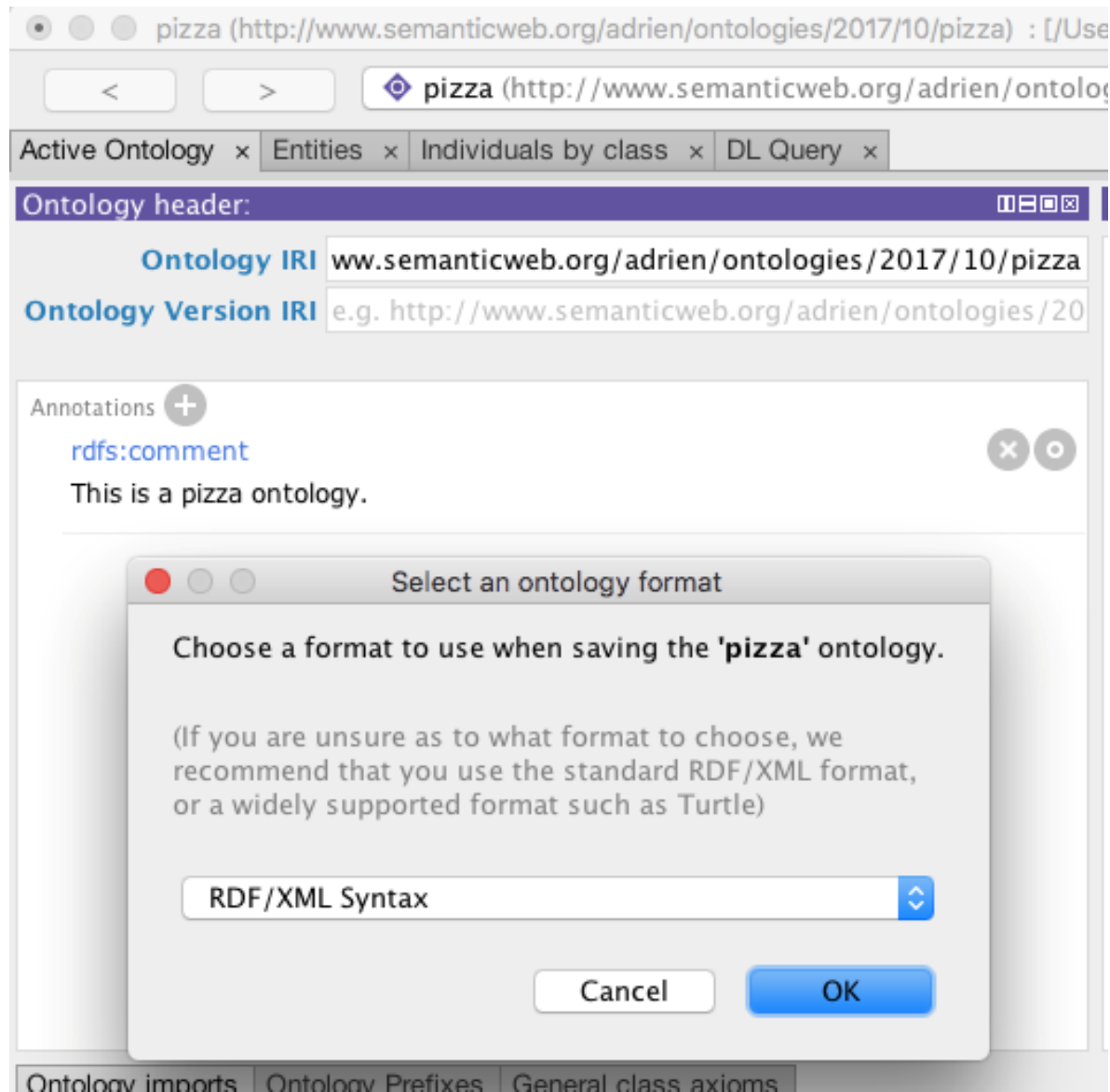


Figure 3.3: Representation Of Classes (Containing Individuals)

Créer une ontologie sous Protégé 5.1






Premières classes

Active Ontology x Entities x Individuals by class x DL Query x

Class hierarchy Class hierarchy (inferred)

Class hierarchy: Pizza



Description: Pizza

Equivalent To +

SubClass Of +

General class axioms +




SubClass Of (Anonymous Ancestor)


Instances +

Target for Key +

Disjoint With +

Class hierarchy Expression editor



```
graph TD; owl:Thing --> Pizza; owl:Thing --> PizzaBase; owl:Thing --> PizzaTopping;
```

Créer des taxonomies de classes

- Sélectionner 'PizzaTopping'; Tools -> Create class hierarchy:

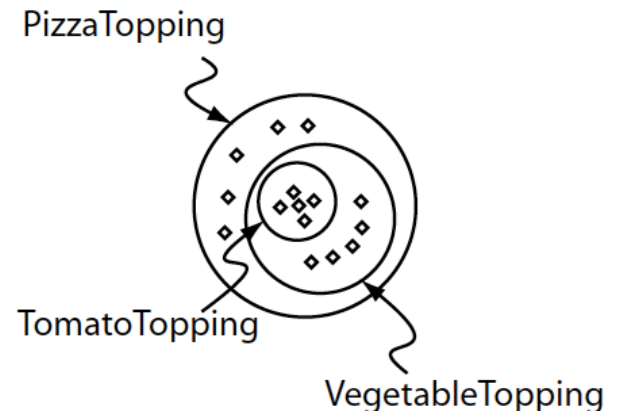
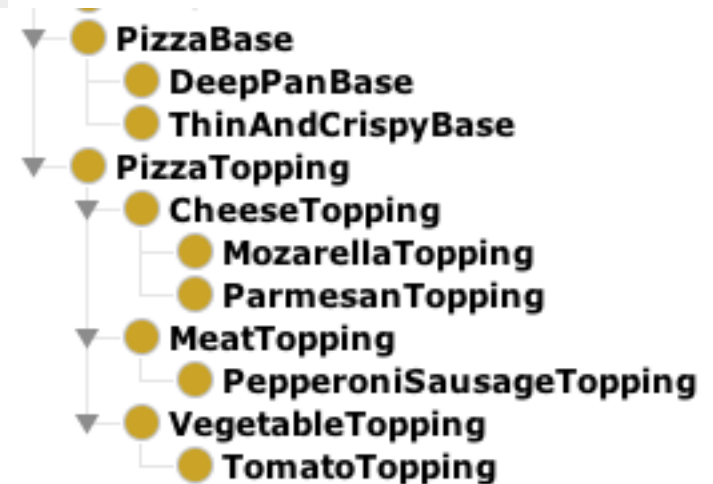
Enter hierarchy

Please enter one name per line. You can use tabs to indent names to create a hierarchy.

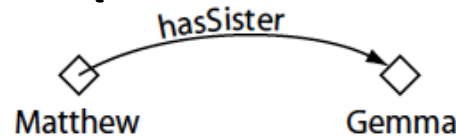
```
Cheese
  Mozzarella
  Parmesan
Meat
  PepperoniSausage
Vegetable
  Tomato
```

Prefix

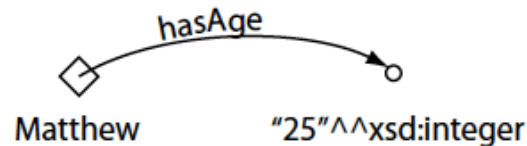
Suffix



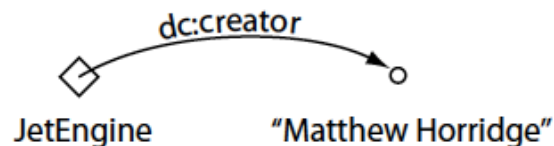
Trois types de propriétés (aka relations)



An object property linking the individual Matthew to the individual Gemma



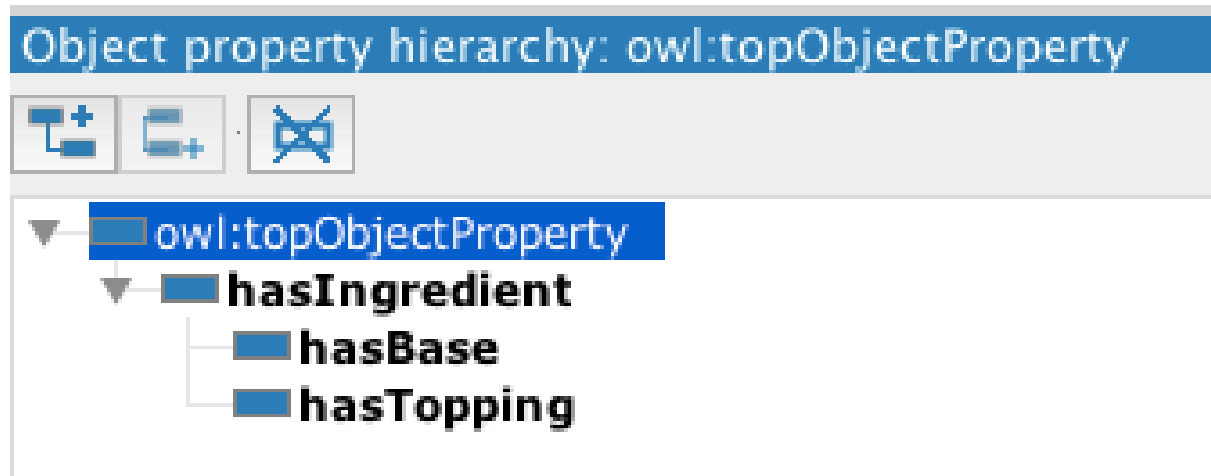
A datatype property linking the individual Matthew to the data literal '25', which has a type of an xsd:integer.



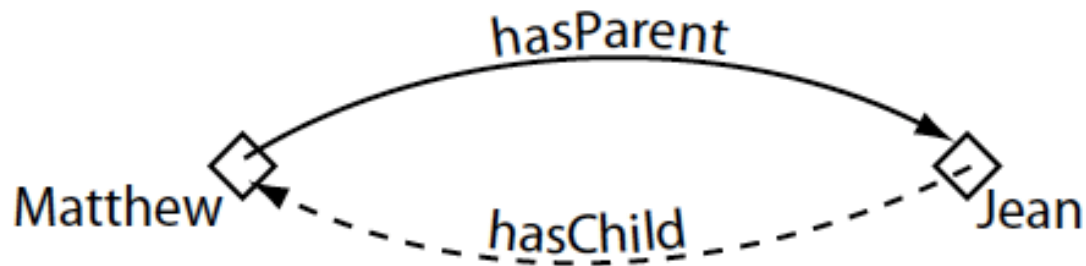
An annotation property, linking the class 'JetEngine' to the data literal (string) "Matthew Horridge".

Figure 4.12: The Different types of OWL Properties

Créer des 'object properties'



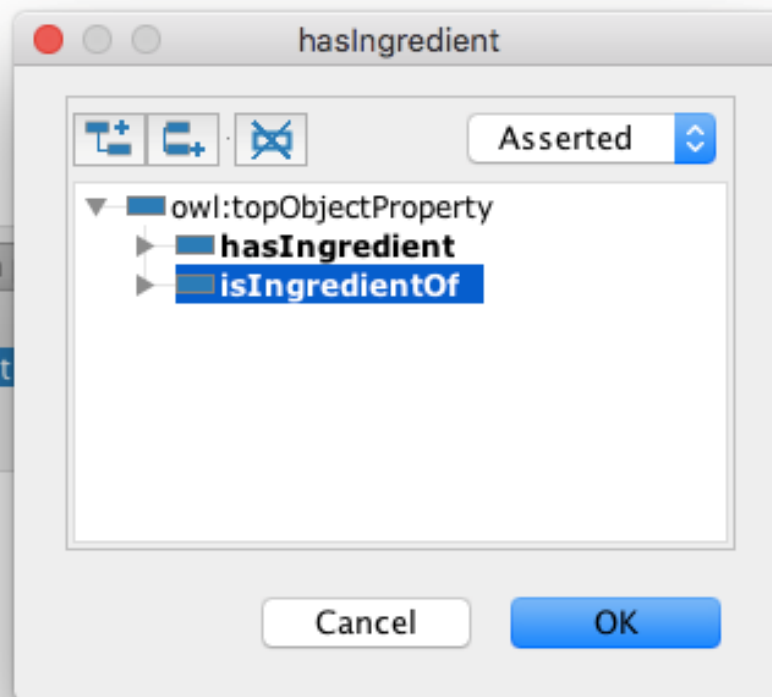
Relations inverses



Individuals by type Annotation
Object property hierarchy
Object property hierarchy: hasIngredient



owl:topObjectProperty
 isIngredientOf
 isToppingOf
 isBaseOf
 hasIngredient
 hasBase
 hasTopping



Description: hasIngredient

Equivalent To +

SubProperty Of +

Inverse Of +

Domains (intersection) +

Ranges (intersection) +

Disjoint With +

SuperProperty Of (Chain) +

Caractéristique des relations

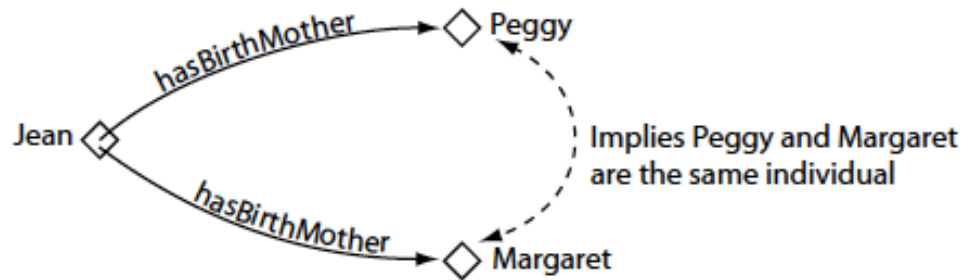


Figure 4.18: An Example Of A Functional Property: `hasBirthMother`

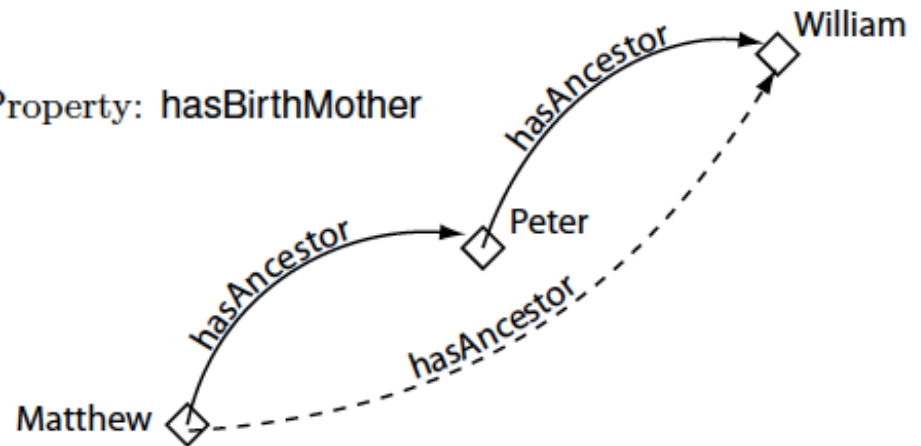


Figure 4.20: An Example Of A Transitive Property: `hasAncestor`

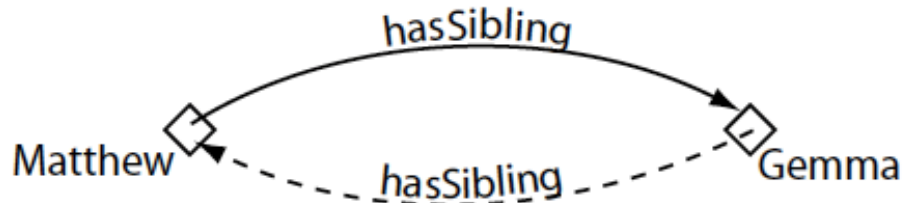


Figure 4.21: An Example Of A Symmetric Property: `hasSibling`

Caractéristiques des relations : application

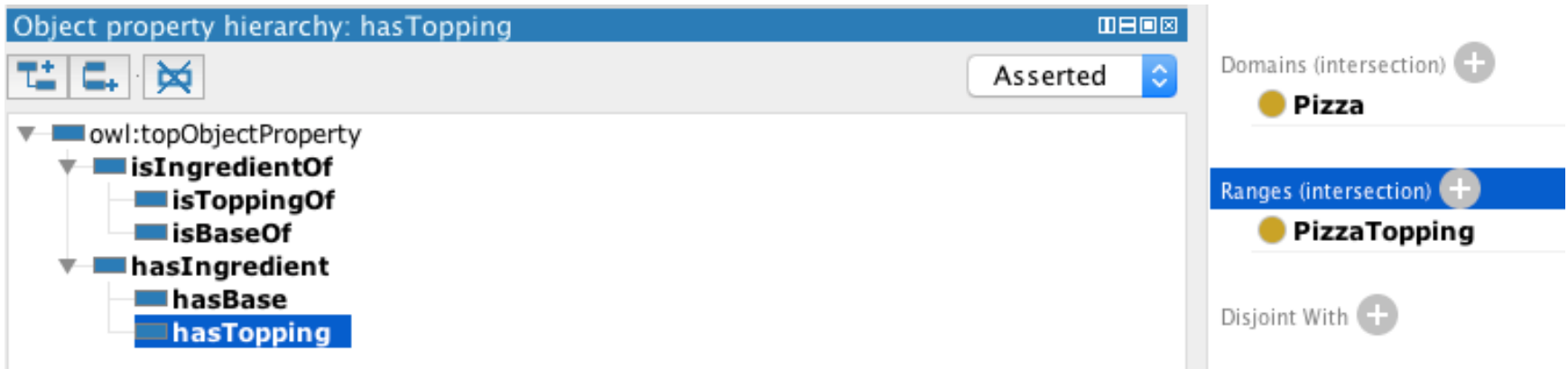
- Rendre 'hasIngredient' et 'isIngredientOf' transitives
- Rendre 'hasBase' fonctionnelle
- Veut-on 'hasTopping' fonctionnelle ou symétrique ?

The screenshot shows a software interface with a tabbed menu at the top containing 'Individuals by type', 'Annotation property hierarchy', and 'Datatypes'. Below this, there are two sub-tabs: 'Object property hierarchy' and 'Data property hierarchy'. The 'Object property hierarchy' tab is active, displaying a tree structure under the heading 'Object property hierarchy: hasIngredient'. The tree shows a hierarchy starting with 'owl:topObjectProperty', followed by 'isIngredientOf', which branches into 'isToppingOf' and 'isBaseOf'. 'isToppingOf' further branches into 'hasIngredient' and 'hasBase'. 'hasIngredient' branches into 'hasBase' and 'hasTopping'. The 'hasIngredient' node is highlighted with a blue background. To the right of the tree, there is a button labeled 'Asserted' with a dropdown arrow.

The screenshot shows a panel titled 'Characteristics: hasIngredient' with two tabs: 'Description' and 'Characteristics'. The 'Characteristics' tab is active, displaying a list of checkboxes for the property's characteristics. The 'Transitive' checkbox is checked, while the others are unchecked.

Characteristic	Checked
Functional	<input type="checkbox"/>
Inverse functional	<input type="checkbox"/>
Transitive	<input checked="" type="checkbox"/>
Symmetric	<input type="checkbox"/>
Asymmetric	<input type="checkbox"/>
Reflexive	<input type="checkbox"/>
Irreflexive	<input type="checkbox"/>

Domain & Range



hasBase: Domain Pizza, Range PizzaBase (utiliser autocomplete)



Property Domains And Ranges In OWL — It is important to realise that in OWL domains and ranges should *not* be viewed as constraints to be checked. They are used as ‘axioms’ in reasoning. For example if the property **hasTopping** has the domain set as **Pizza** and we then applied the **hasTopping** property to **IceCream** (individuals that are members of the class **IceCream**), this would generally not result in an error. It would be used to infer that the class **IceCream** must be a subclass of **Pizza**! ^a.


^aAn error will only be generated (by a reasoner) if **Pizza** is disjoint to **IceCream**


Axiomes (aka restrictions) :


Restrictions existentielles

Créer NamedPizza sous-classe de Pizza


Description: Pizza


Equivalent To 





SubClass Of 


-  **hasBase some PizzaBase**

Description: AmericanaPizza


Equivalent To 

SubClass Of 


-  **hasTopping some MozarellaTopping**
-  **hasTopping some PepperoniSausageTopping**
-  **hasTopping some TomatoTopping**
-  **NamedPizza**


General class axioms 




SubClass Of (Anonymous Ancestor)


-  **hasBase some PizzaBase**

Description: MargheritaPizza


Equivalent To 

SubClass Of 

-  **hasTopping some MozarellaTopping**
-  **hasTopping some TomatoTopping**
-  **NamedPizza**

General class axioms 

SubClass Of (Anonymous Ancestor)

-  **hasBase some PizzaBase**

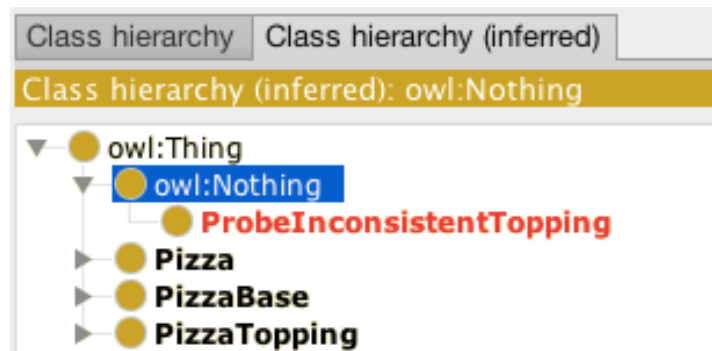
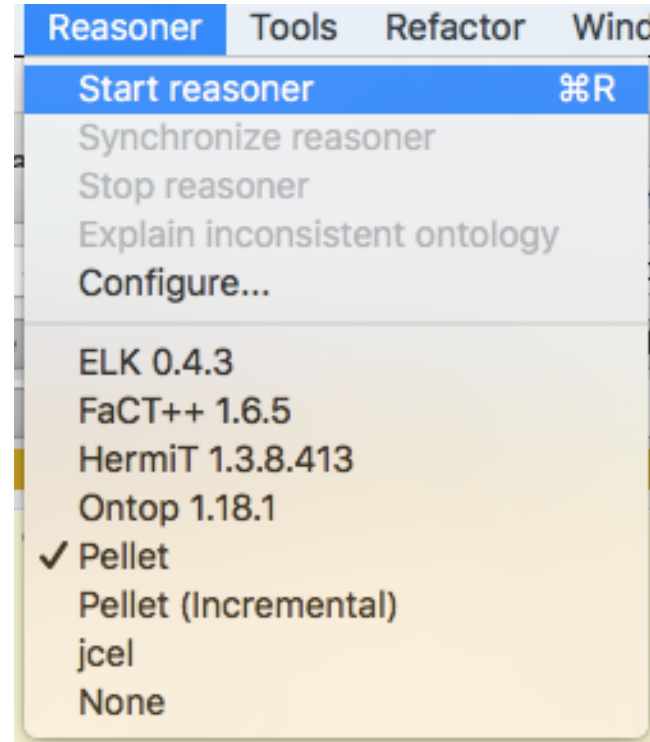
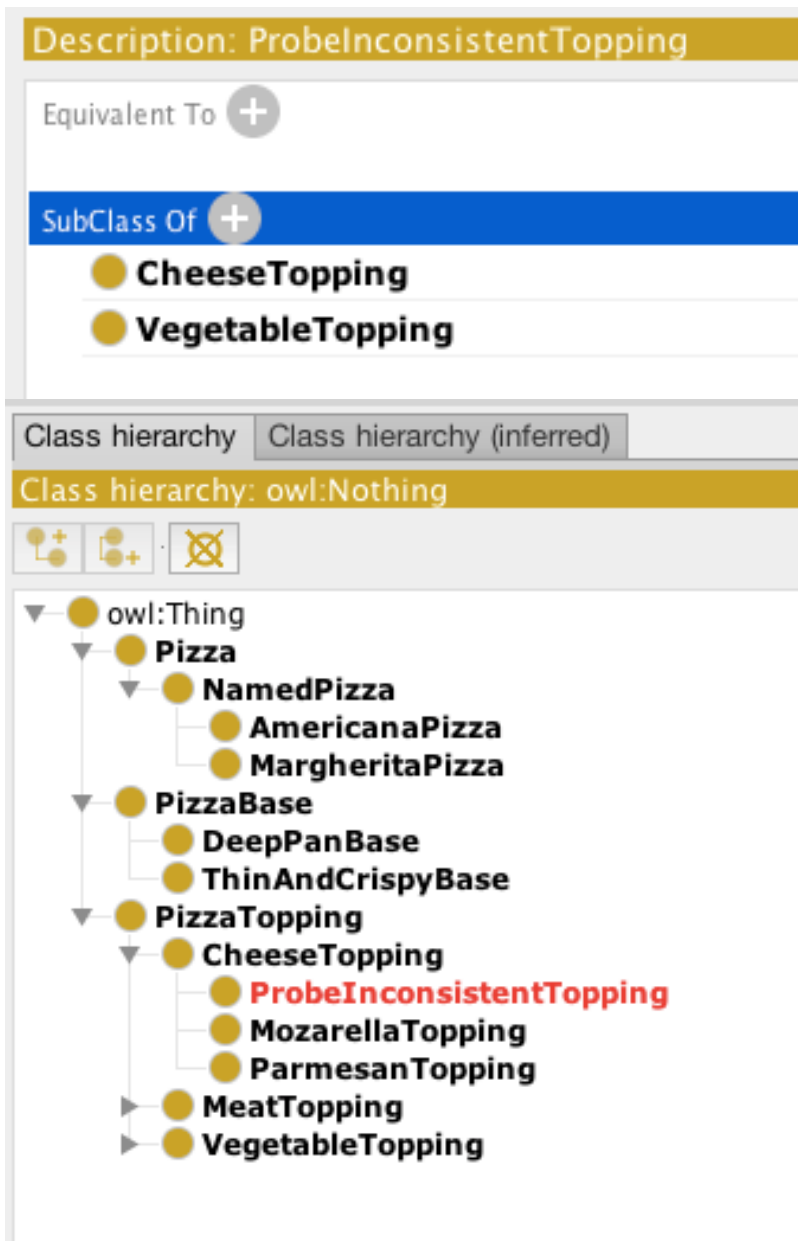
Pour créer AmericanaPizza:

- Sélectionner MargheritaPizza
- Edit / Duplicate selected class
- Changer nom pour 'AmericanaPizza'
- Cliquer OK
- Ajouter axiome 'hasTopping some PepperoniSausageTopping'

Rendre MargheritaPizza et AmericanaPizza disjointes

Remarque : Une restriction définit une classe anonyme.

Raisonneur : détecter les incohérences



Ne pas oublier d'arrêter le raisonneur.
Réessayer en enlevant l'axiome de disjointure de
CheeseTopping et VegetableTopping.

Classes primitive et classes définies (aka classes équivalentes)



A class that only has *necessary* conditions is known as a **Primitive Class**.

Créer CheesyPizza

Description: CheesyPizza

Equivalent To +

SubClass Of +

● hasTopping **some** CheeseTopping

● Pizza



A class that has at least one set of *necessary and sufficient* conditions is known as a **Defined Class**.

Edit -> Convert to defined class

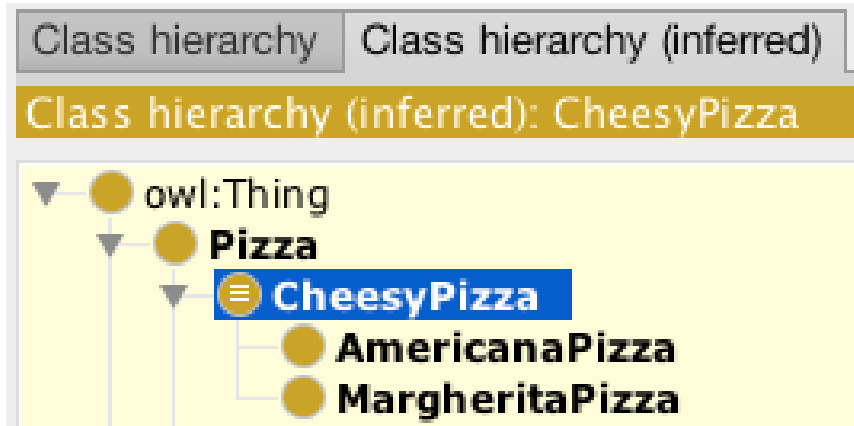
Description: CheesyPizza

Equivalent To +

● Pizza

and (hasTopping **some** CheeseTopping)

Raisonneur : classification automatique



- Héritage multiple := Certaines classes ont plusieurs classes parentes.
- Principe méthodologique: Construire une hiérarchie à héritage simple asserté.
- Le raisonneur pourra inférer et maintenir l'héritage multiple.





It is important to realise that, in general, classes will never be placed as sub-classes of *primitive* classes (i.e. classes that only have necessary conditions) by the reasoner^a.

^aThe exception to this is when a property has a domain that is a primitive class. This can *coerce* classes to be reclassified under the primitive class that is the domain of the property — the use of property domains to cause such effects is strongly discouraged.

Restrictions universelles

Description: VegetarianPizza

Equivalent To 


SubClass Of 

- hasTopping **only** (CheeseTopping **or** VegetableTopping)
- Pizza

Voyez-vous un problème avec cette caractérisation ?

Edit -> Convert to defined class

Description: VegetarianPizza

Equivalent To 

- Pizza
and (hasTopping **only**
(CheeseTopping **or** VegetableTopping))

Raisonnement en monde ouvert

- MargheritaPizza sera-t-elle classée en sous-classe de VegetarianPizza ?
- Hypothèse de **monde ouvert** : on ne peut pas supposer que quelque chose n'existe pas à moins qu'il ne soit explicitement énoncé qu'il n'existe pas.
- Pour que MargheritaPizza soit classée en sous-classe de VegetarianPizza, il faut ajouter un axiome de fermeture.

Description: MargheritaPizza

Equivalent To 

SubClass Of 

 hasTopping **only** (MozarellaTopping  TomatoTopping)

 hasTopping **some** MozarellaTopping

 hasTopping **some** TomatoTopping

 NamedPizza

Remarque : Les axiomes existentiels sont également importants, pour éviter qu'une pizza sans garniture puisse être considérée comme MargheritaPizza.

Clic droit sur axiome existentiel -> 'Create closure axiom'

Restrictions de cardinalité

Créer InterestingPizza sous-classe de Pizza

Description: InterestingPizza

Equivalent To +

● **Pizza**
and (hasTopping **min** 3 PizzaTopping)

Description: FourCheesePizza

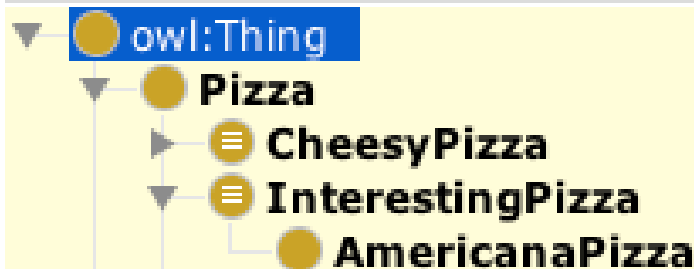
Equivalent To +

SubClass Of +

● hasTopping **exactly** 4 CheeseTopping
● NamedPizza

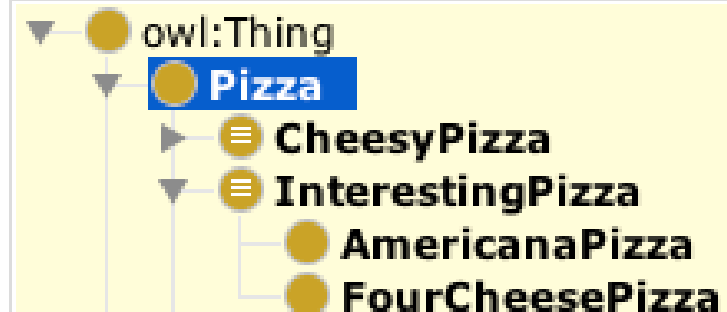
Class hierarchy Class hierarchy (inferred)

Class hierarchy (inferred): owl:Thing



Class hierarchy Class hierarchy (inferred)

Class hierarchy (inferred): Pizza



Datatype properties et individus

Individuals by type | Annotation property hierarchy | Datatypes

Object property hierarchy | Data property hierarchy

Data property hierarchy: owl:topDataProperty

Asserted

owl:topDataProperty

hasCalorificContentValue

Character

☒ Functional

Individuals by type | Annotation

Object property hierarchy

Individuals by type: Example-Margherita

Example-Margherita

Description: Example-Margherita

Property assertions: Example-Margherita

Types

MargheritaPizza

Same Individual As

Different Individuals

Object property assertions

Data property assertions

hasCalorificContent 263

Negative object property assertions

Negative data property assertions

Data Property



Asserted

owl:topDataProperty
hasCalorificContentValue

Value

263

Type

xsd:integer


Lang


Cancel



OK

Axiome : Toutes les pizzas ont une valeur calorifique

Description: Pizza

Equivalent To 





SubClass Of 

-  **hasBase** **some** **PizzaBase**
-  **hasCalorificContentValue** **some** **xsd:integer**

Pizza

Data restriction creator | Class expression editor | Class hierarchy | Object restriction creator















Restricted property

   Asserted 



owl:topDataProperty

hasCalorificContentValue

Restriction filler

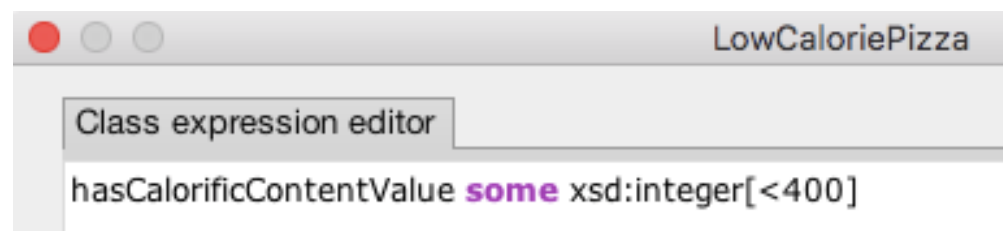
-  xsd:boolean
-  xsd:byte
-  xsd:dateTime
-  xsd:dateTimeStamp
-  xsd:decimal
-  xsd:double
-  xsd:float
-  xsd:hexBinary
-  xsd:int
-  **xsd:integer**
-  xsd:language
-  xsd:long
-  xsd:Name
-  xsd:NCName

Restriction type


Some (existential)  **Cardinality** 1 


Cancel OK


Raisonner avec des datatype properties





Description: LowCaloriePizza

Equivalent To 



-  **Pizza**
- and** (hasCalorificContentValue **some** xsd:integer[< 400])


SubClass Of 


-  **Pizza**

General class axioms 

SubClass Of (Anonymous Ancestor)

-  **hasCalorificContentValue** **some** xsd:integer
-  **hasBase** **some** PizzaBase

Instances 

-  **Example-Margherita**

Analyse ontologique des pays

- On voudrait pouvoir dire que la pizza Margharita a comme pays d'origine l'Italie.
- Les pays sont-ils des classes ou des individus?

Relation entre une classe et un individu

