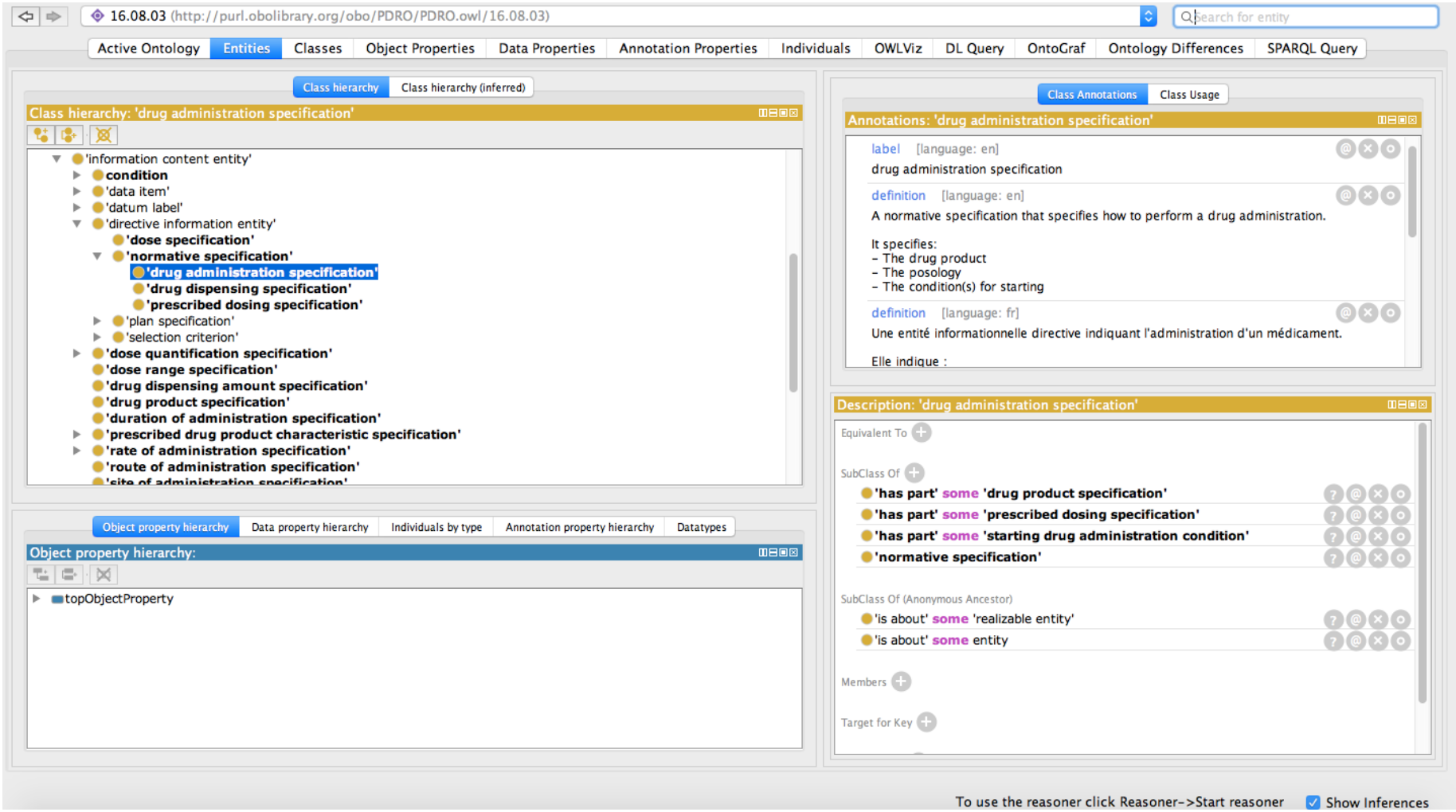
Introduction à OWL et Protégé

École d'été interdisciplinaire en numérique de la santé (EINS 2024)

Adrien Barton[1,2]
Avec la collaboration de Paul Fabry[2]
[1] CNRS, IRIT
[2] GRIIS, Université de Sherbrooke
4 juin 2024


Protégé : un outil de création d'ontologies



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

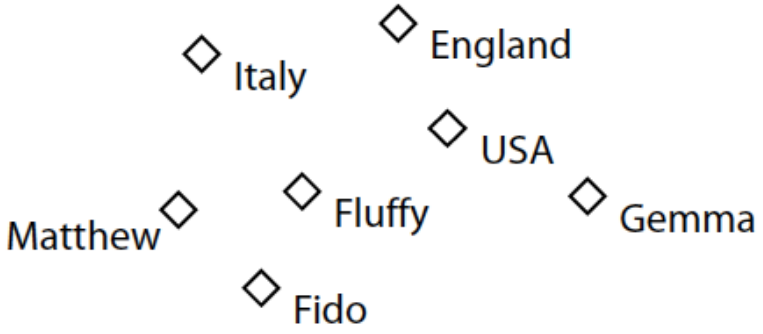


Figure 3.1: Representation Of Individuals

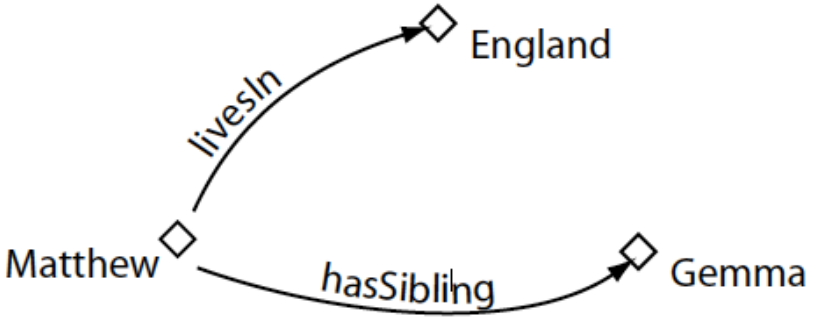


Figure 3.2: Representation Of Properties

Classes

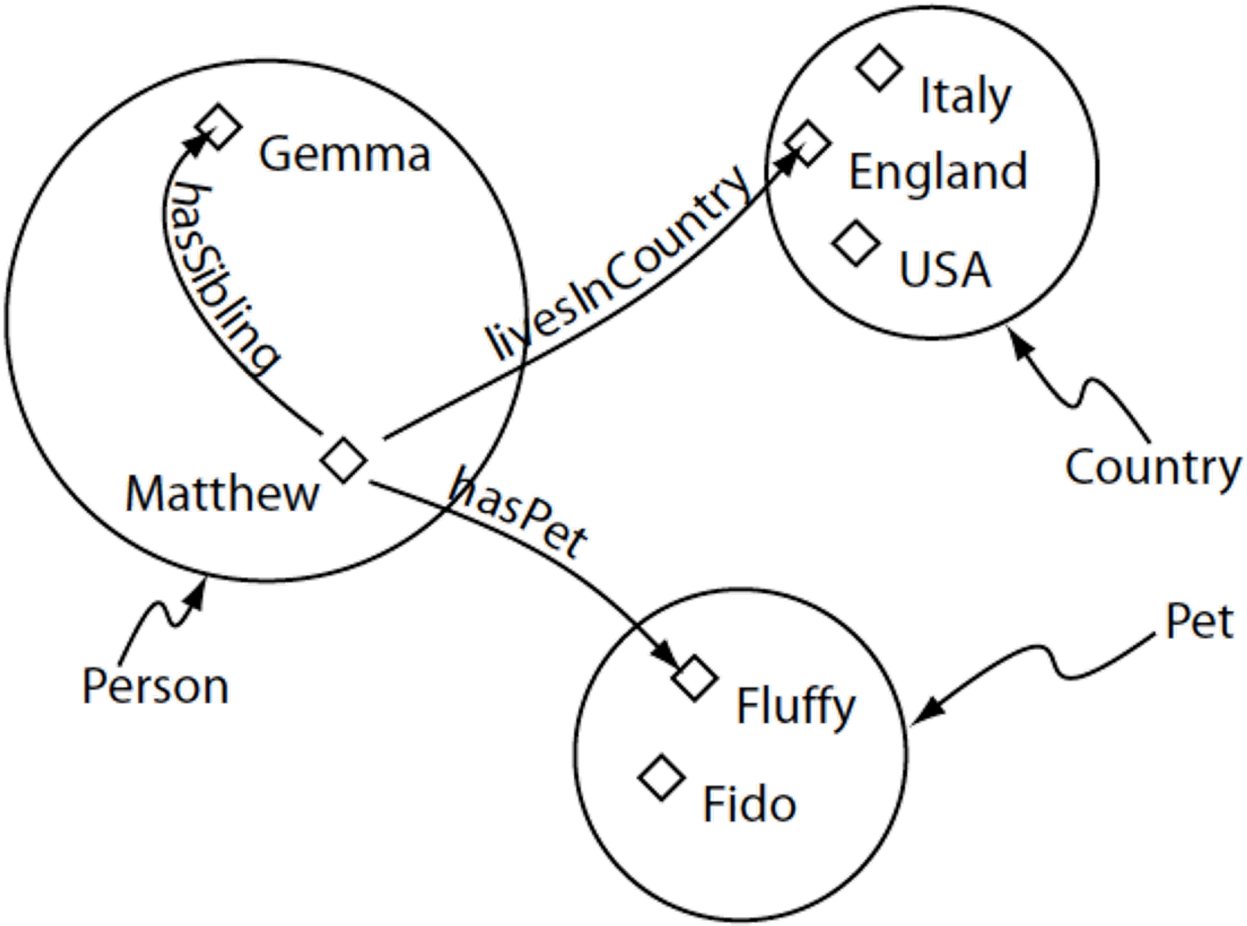
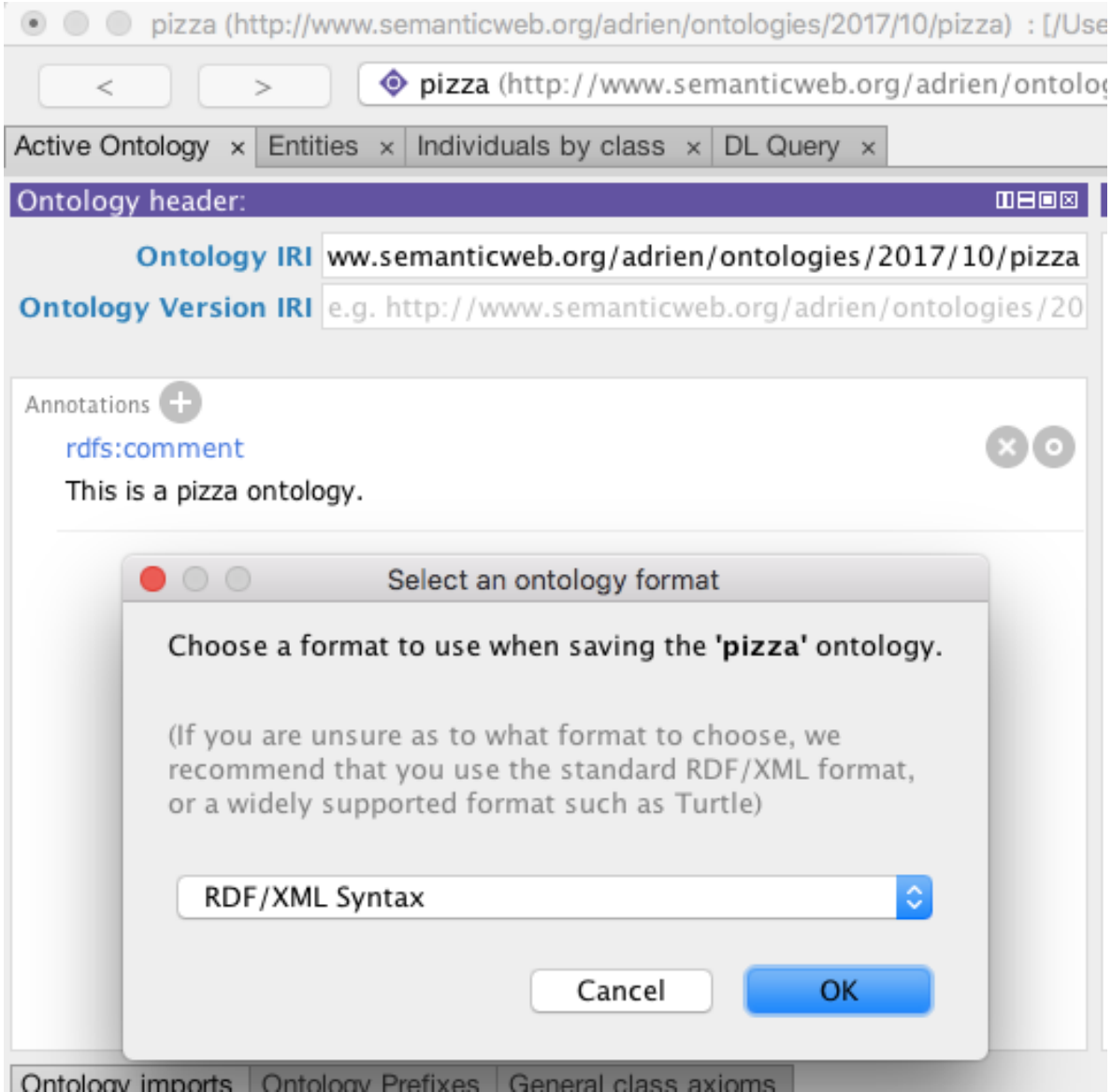
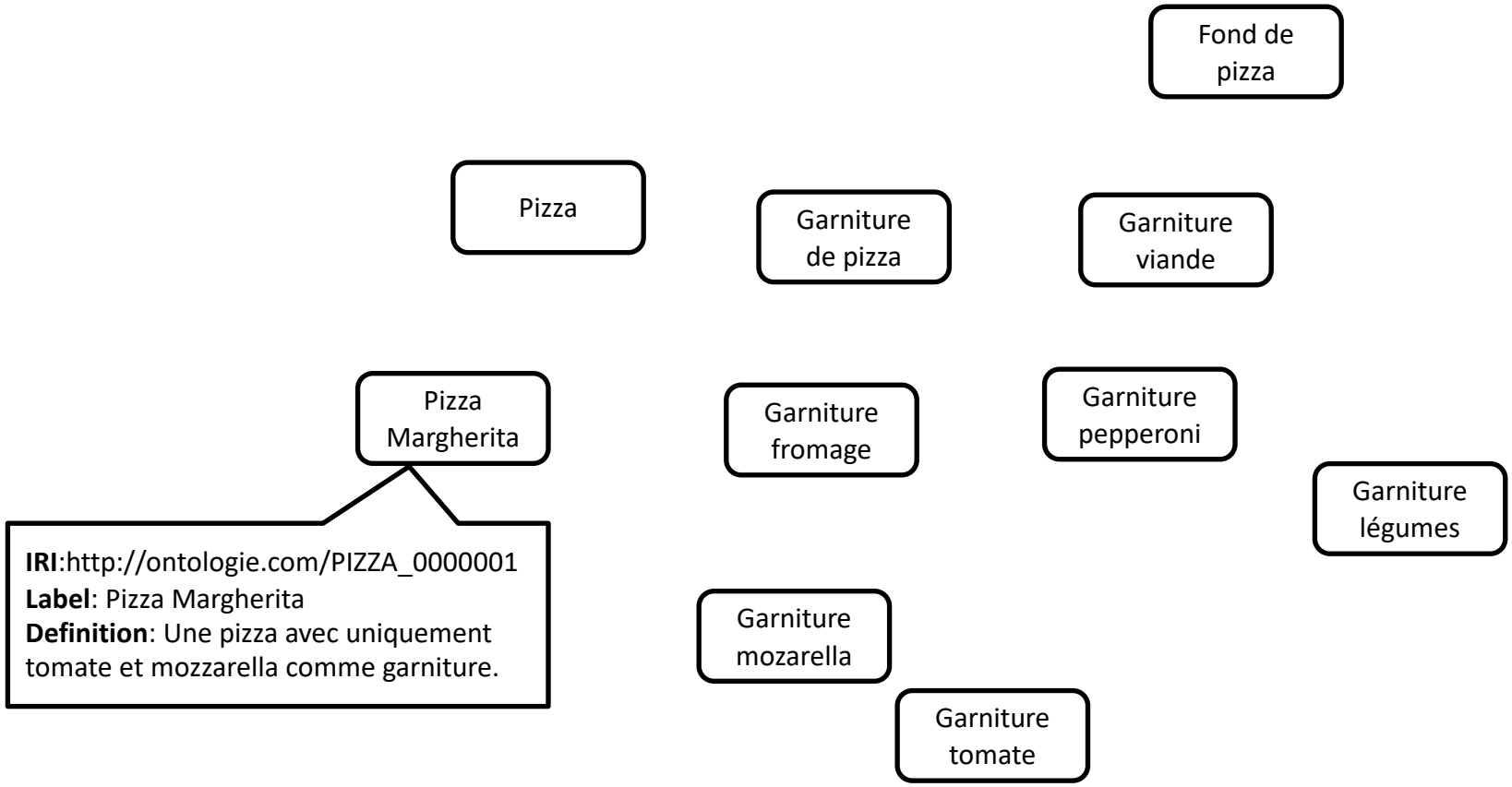


Figure 3.3: Representation Of Classes (Containing Individuals)

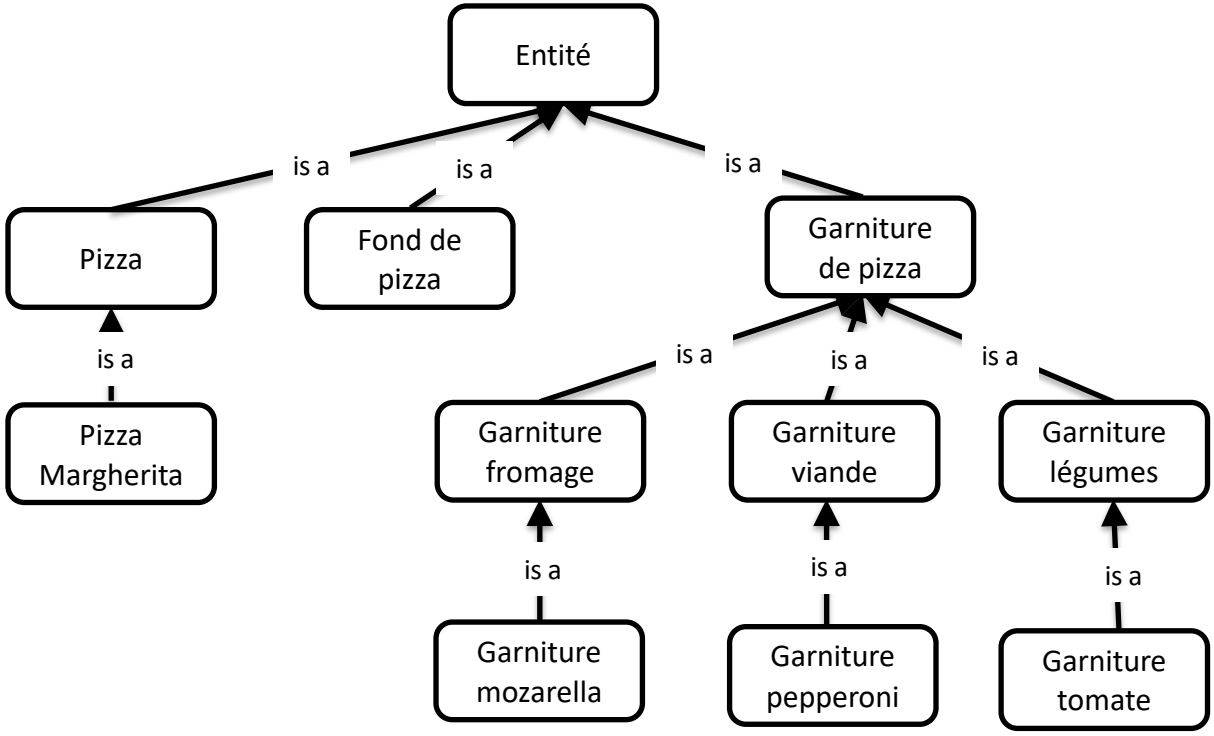
Créer une ontologie sous Protégé 5.x



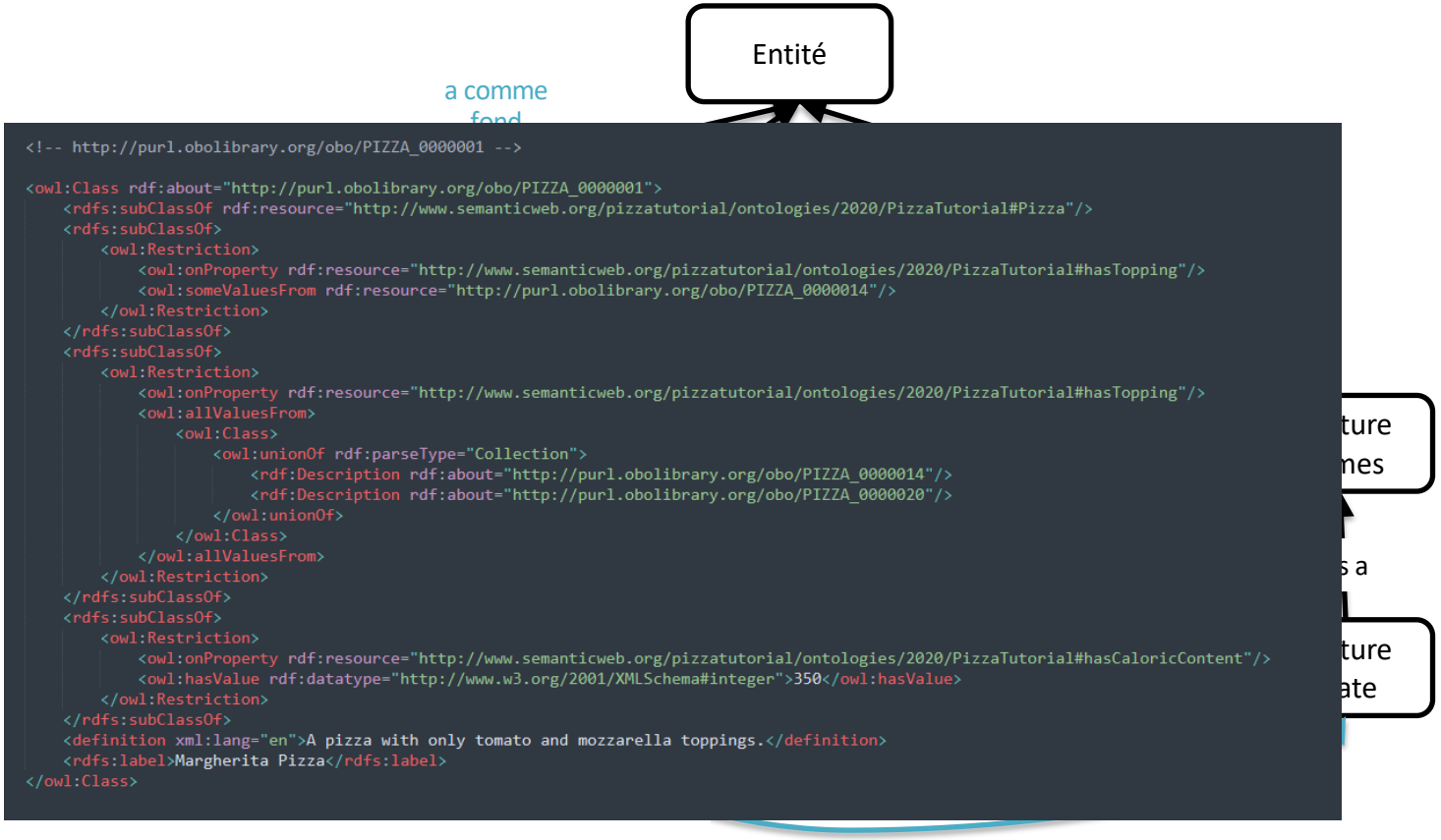
Quelles classes pour une ontologie des pizzas ?



Une première taxonomie des pizzas

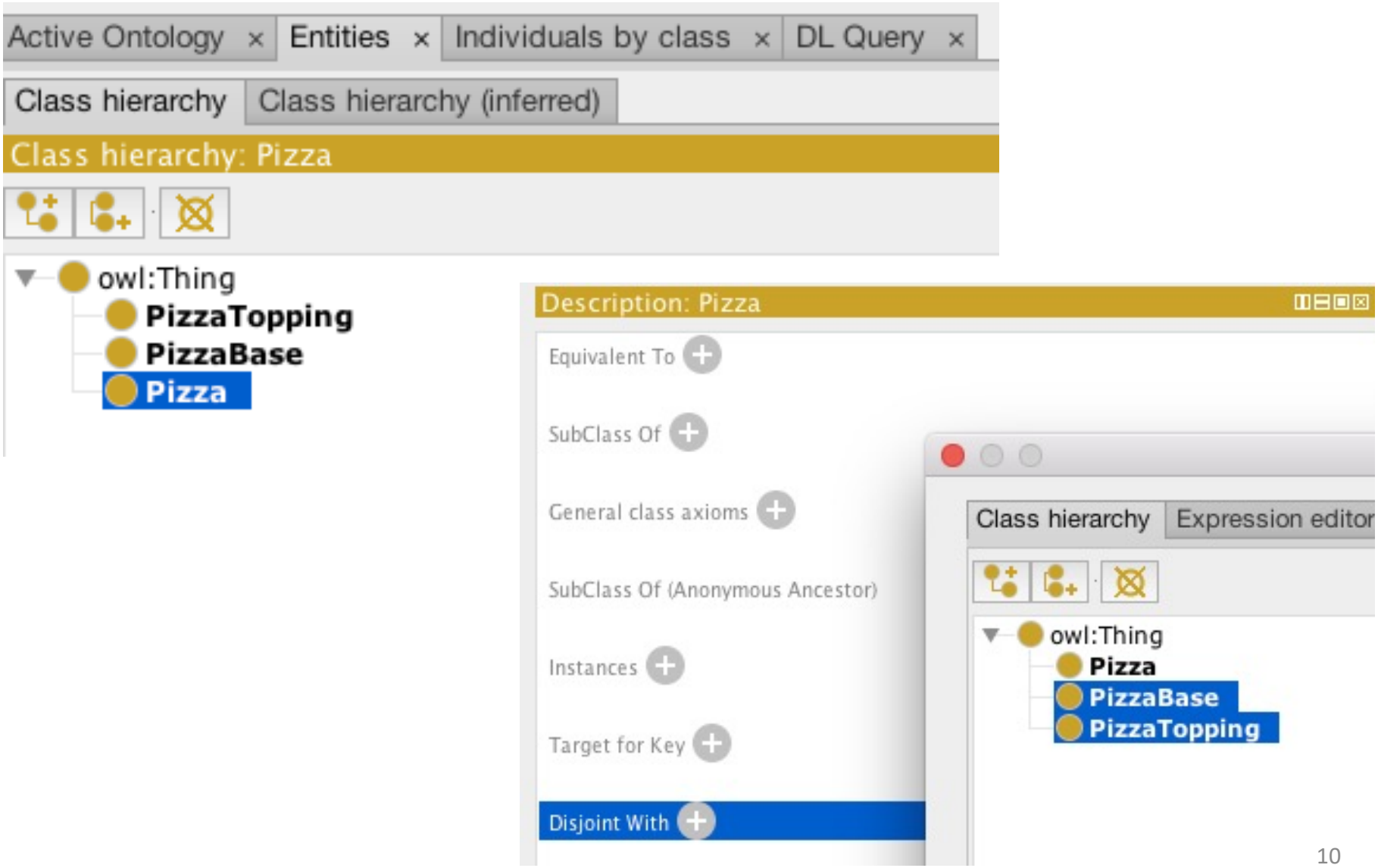


Une première taxonomie des pizzas



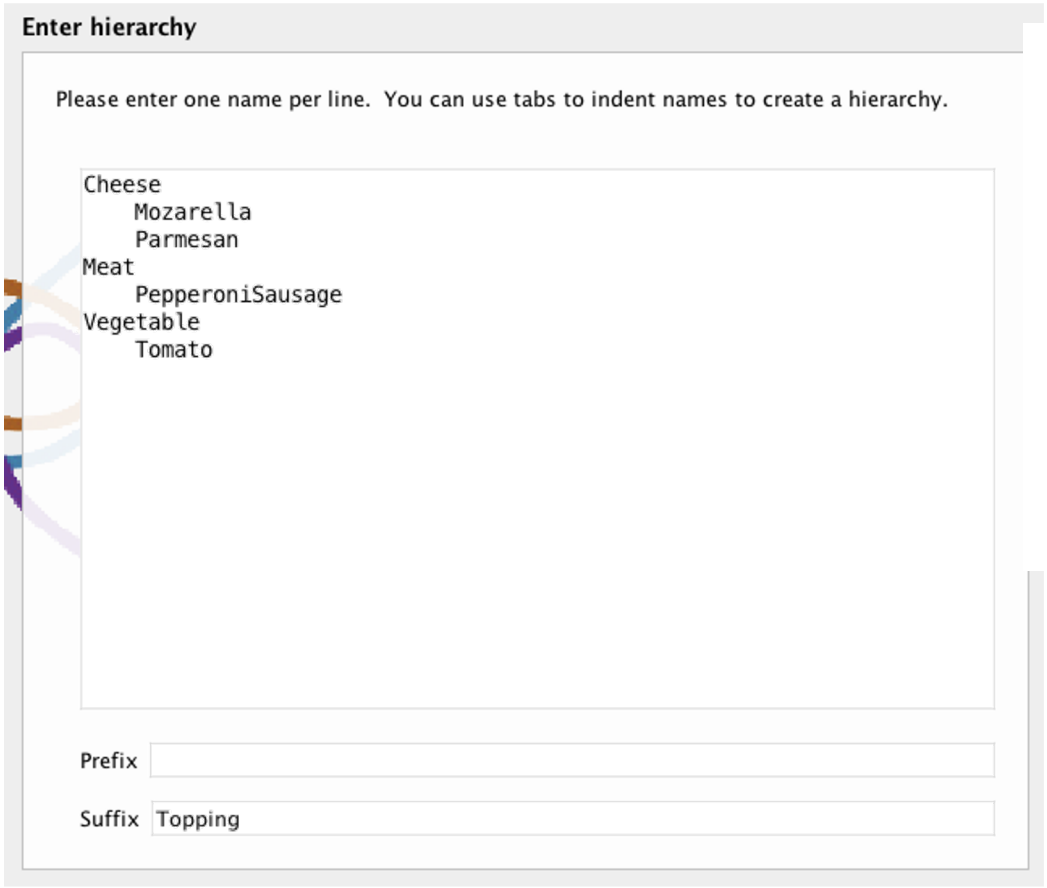

```
<!-- http://purl.obolibrary.org/obo/PIZZA_0000001 -->

<owl:Class rdf:about="http://purl.obolibrary.org/obo/PIZZA_0000001">
    <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/pizzatutorial/ontologies/2020/PizzaTutorial#Pizza"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.semanticweb.org/pizzatutorial/ontologies/2020/PizzaTutorial#hasTopping"/>
            <owl:someValuesFrom rdf:resource="http://purl.obolibrary.org/obo/PIZZA_0000014"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.semanticweb.org/pizzatutorial/ontologies/2020/PizzaTutorial#hasTopping"/>
            <owl:allValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="http://purl.obolibrary.org/obo/PIZZA_0000014"/>
                        <rdf:Description rdf:about="http://purl.obolibrary.org/obo/PIZZA_0000020"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:allValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.semanticweb.org/pizzatutorial/ontologies/2020/PizzaTutorial#hasCaloricContent"/>
            <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">350</owl:hasValue>
        </owl:Restriction>
    </rdfs:subClassOf>
    <definition xml:lang="en">A pizza with only tomato and mozzarella toppings.</definition>
    <rdfs:label>Margherita Pizza</rdfs:label>
</owl:Class>
```

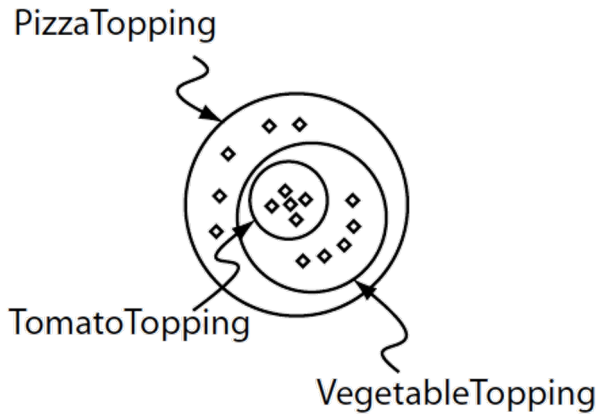
Premières classes



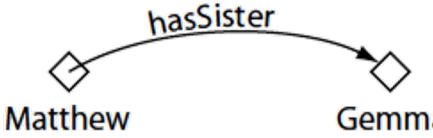
Créer des taxonomies de classes

- Sélectionner ‘PizzaTopping’; Tools -> Create class hierarchy:



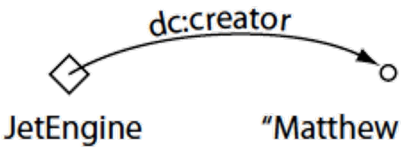


Trois types de propriétés (aka relations)



An object property linking the individual Matthew to the individual Gemma

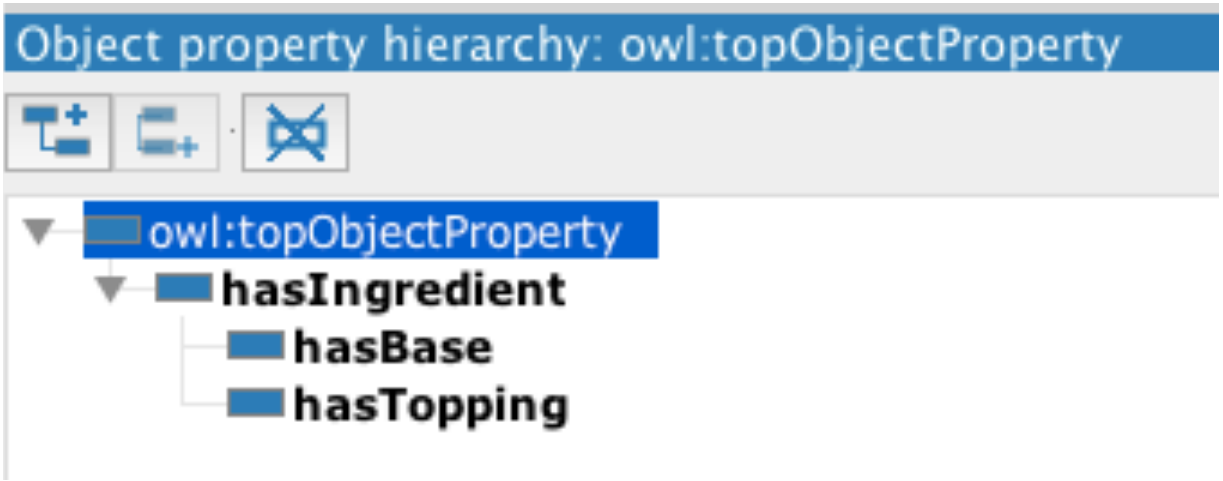
A datatype property linking the individual Matthew to the data literal '25', which has a type of an xsd:integer.



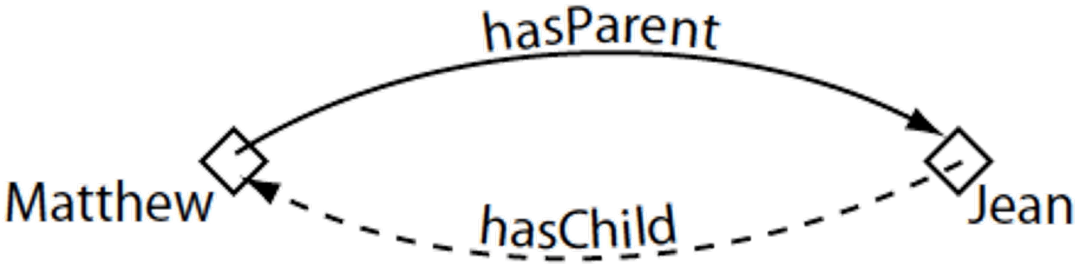
An annotation property, linking the class 'JetEngine' to the data literal (string) "Matthew Horridge".

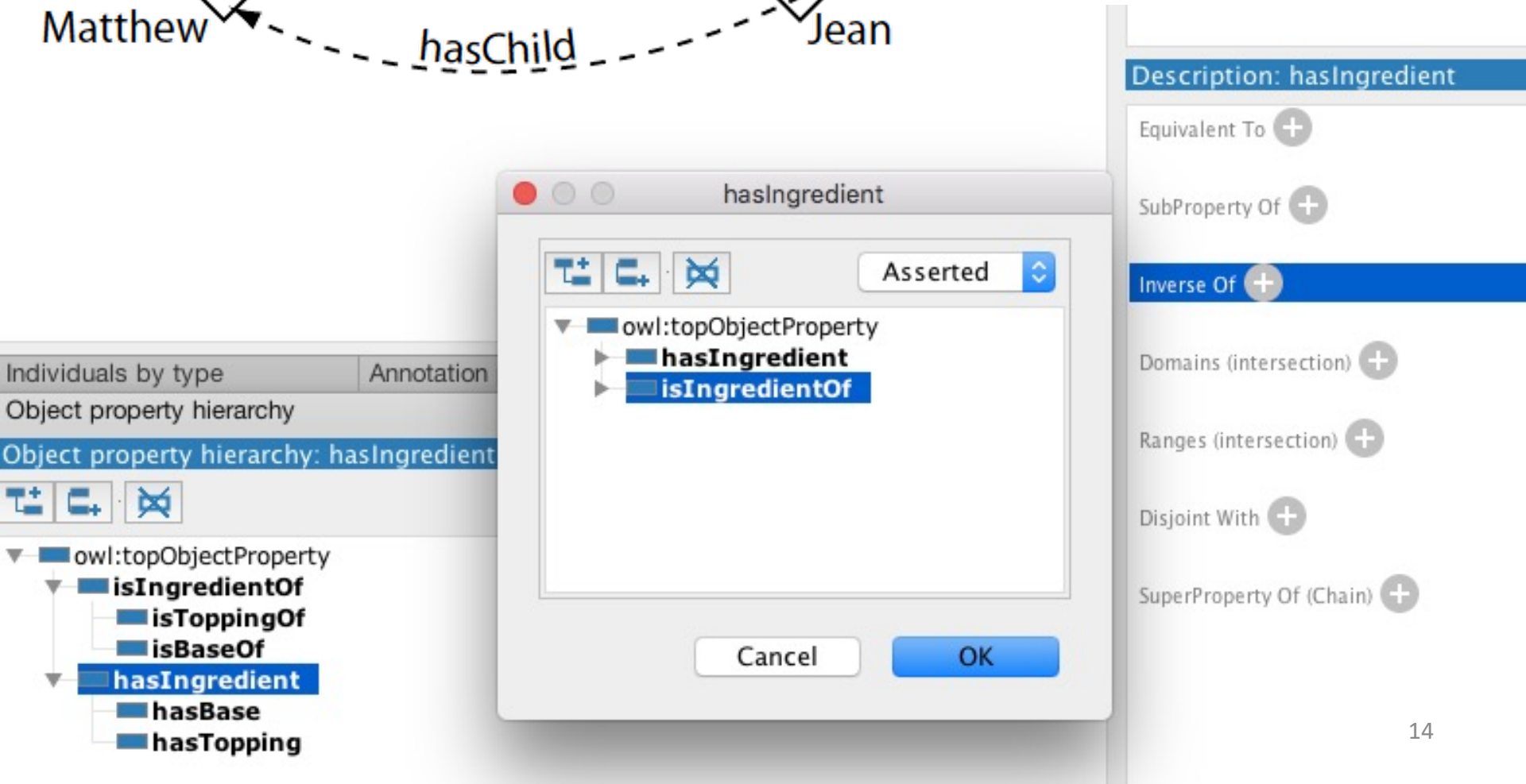
Figure 4.12: The Different types of OWL Properties

Créer des ‘object properties’



Relations inverses





Caractéristique des relations

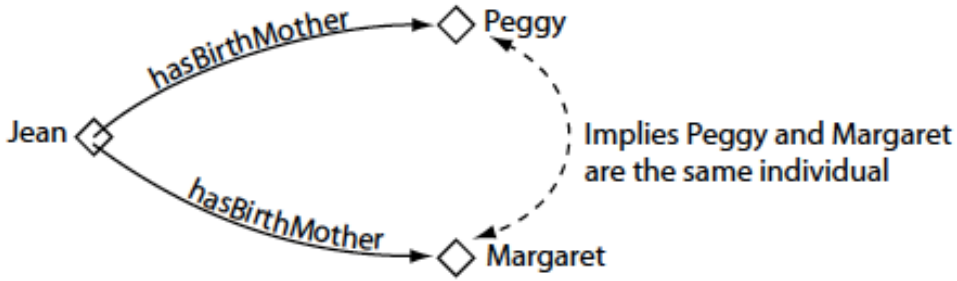


Figure 4.18: An Example Of A Functional Property: hasBirthMother

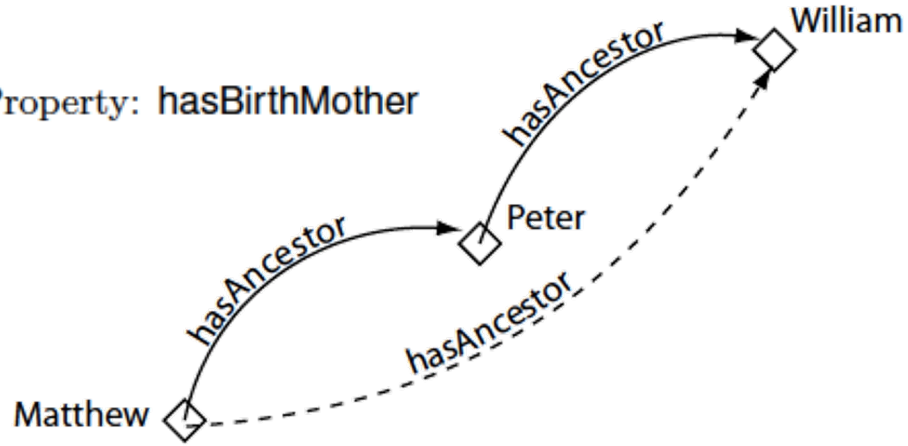


Figure 4.20: An Example Of A Transitive Property: hasAncestor

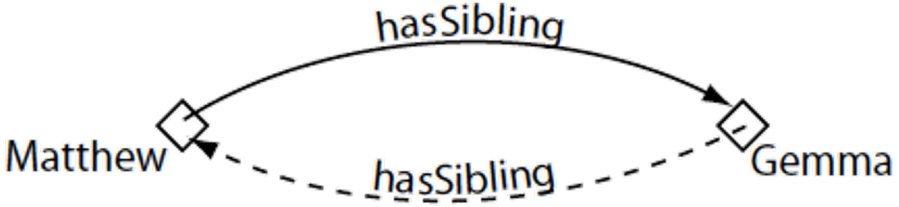
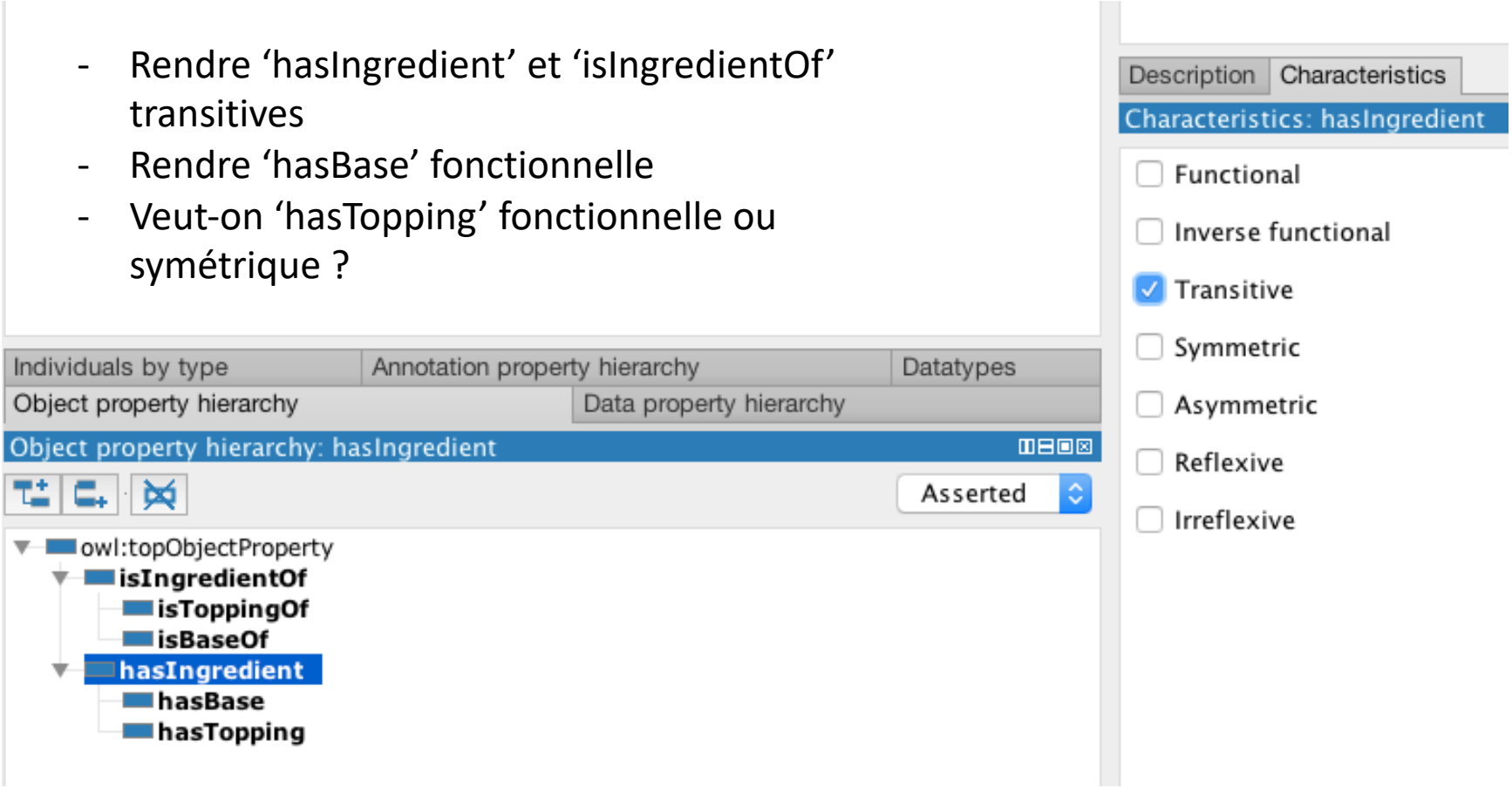


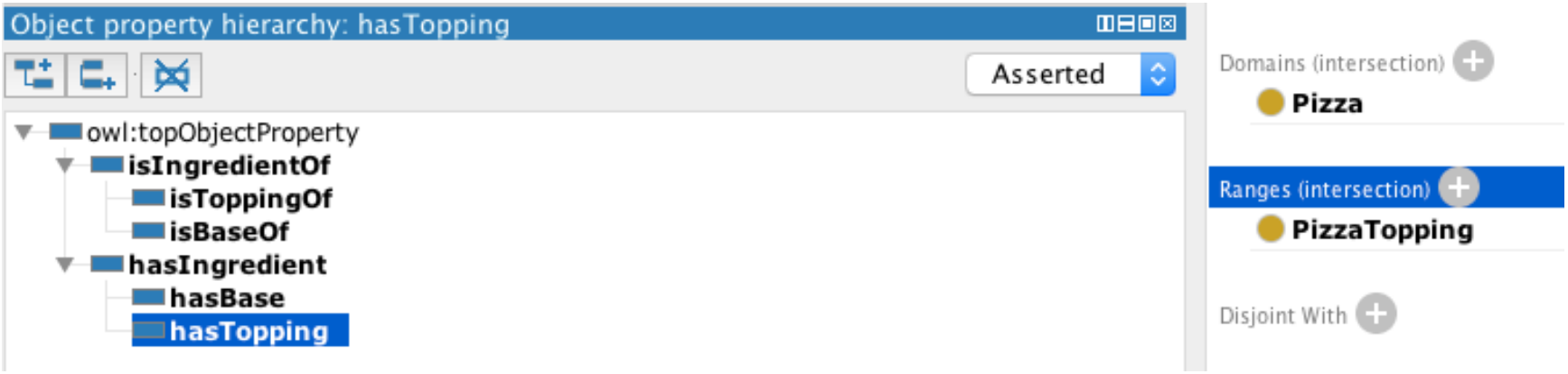
Figure 4.21: An Example Of A Symmetric Property: hasSibling

Caractéristiques des relations : application

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



Domain & Range

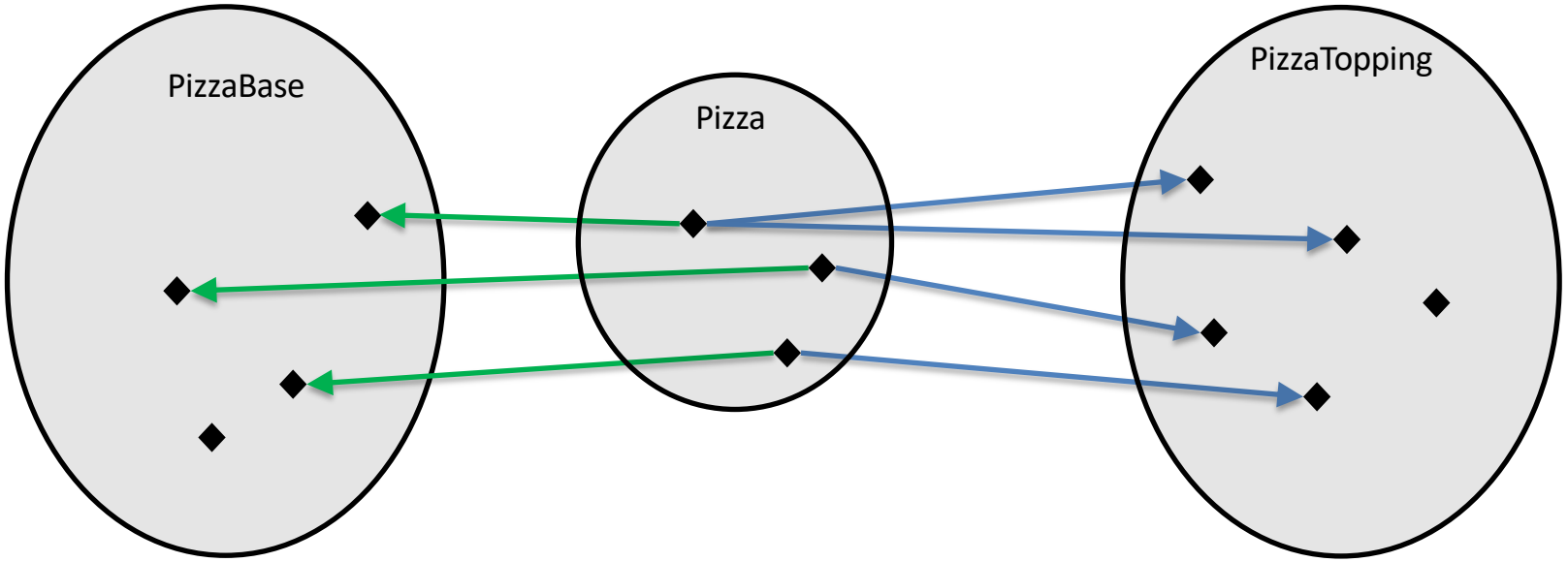


hasBase: Domain Pizza, Range PizzaBase (utiliser autocomplete)

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].

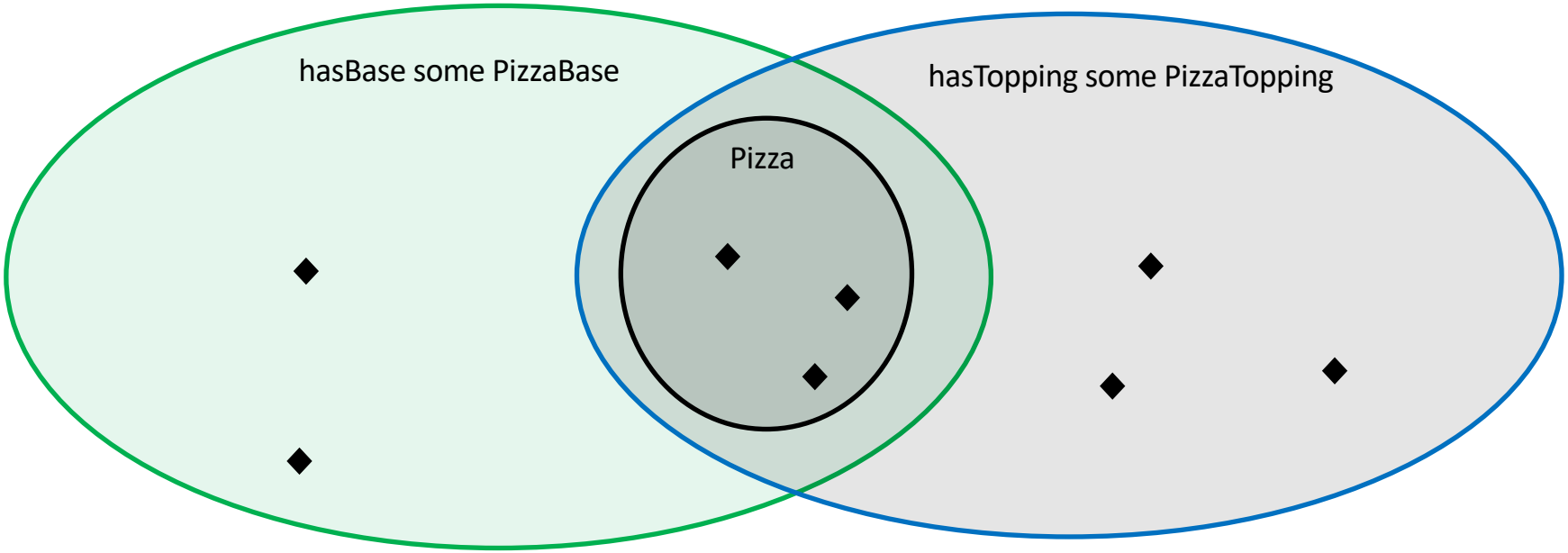
[a] An error will only be generated (by a reasoner) if Pizza is disjoint to IceCream

Axiomes existentiels (restrictions existentielles)



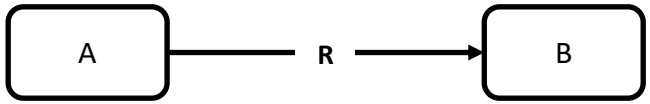
Axiomes existentiels (restrictions existentielles)

A SubclassOf R some B

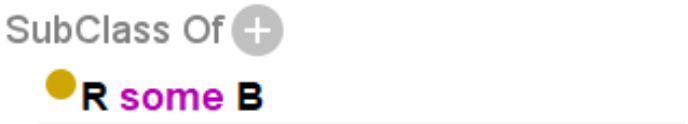


Axiomes existentiels (restrictions existentielles)

Un axiome est une restriction de classe

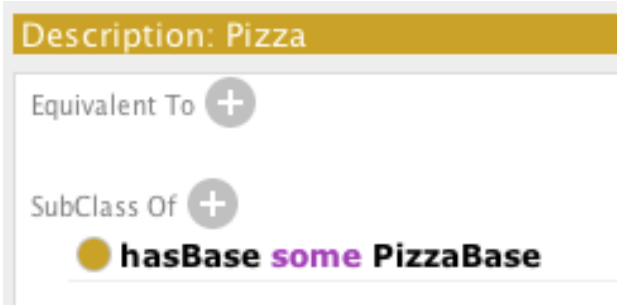


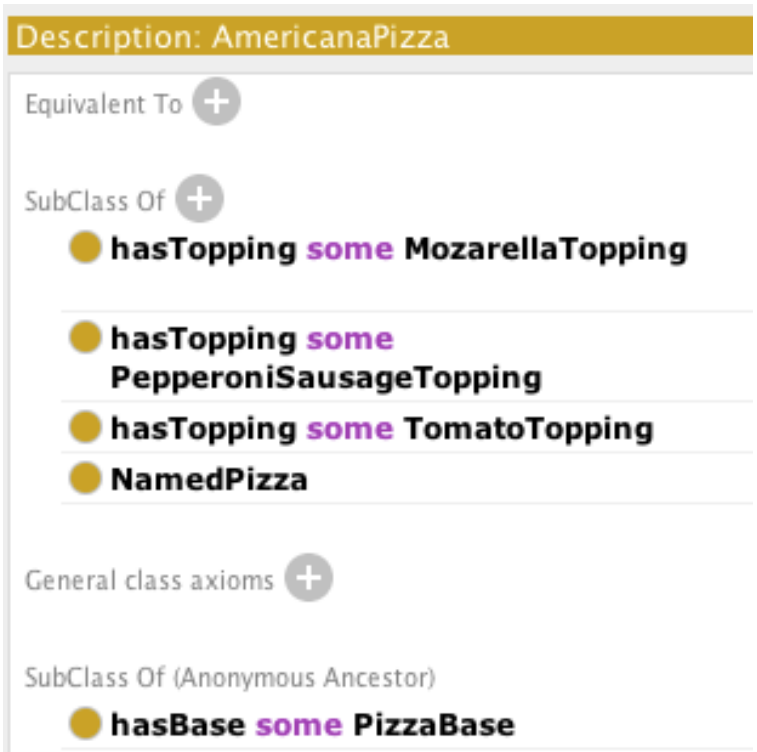
A SubClassOf (R *some* B)

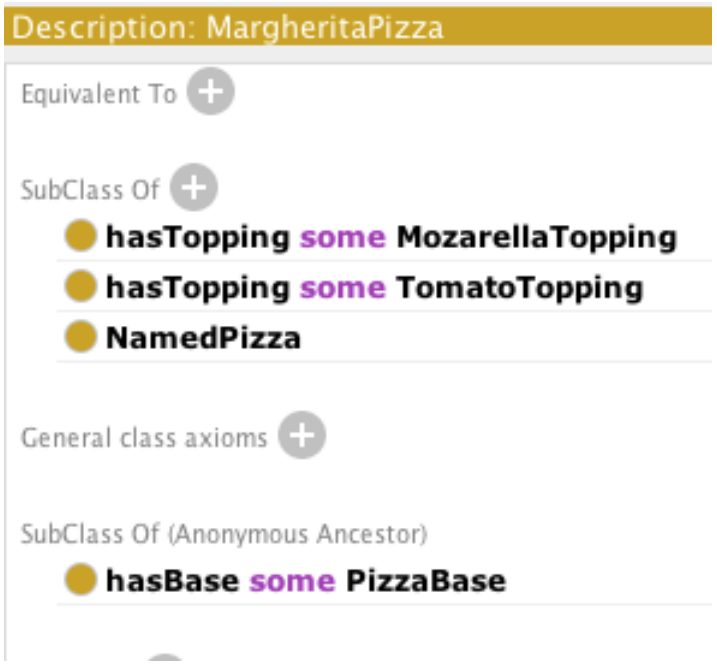


Axiomes existentiels (restrictions existentielles)

Créer NamedPizza sous-classe de Pizza





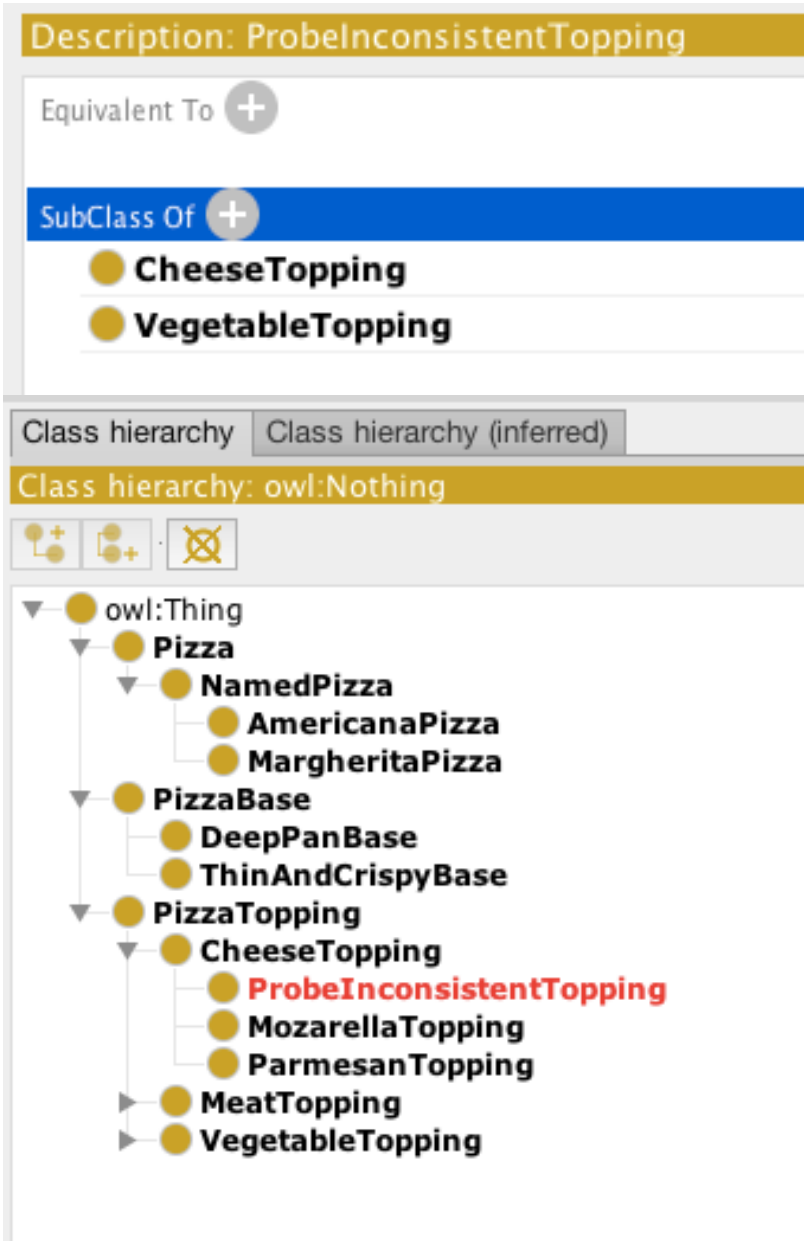


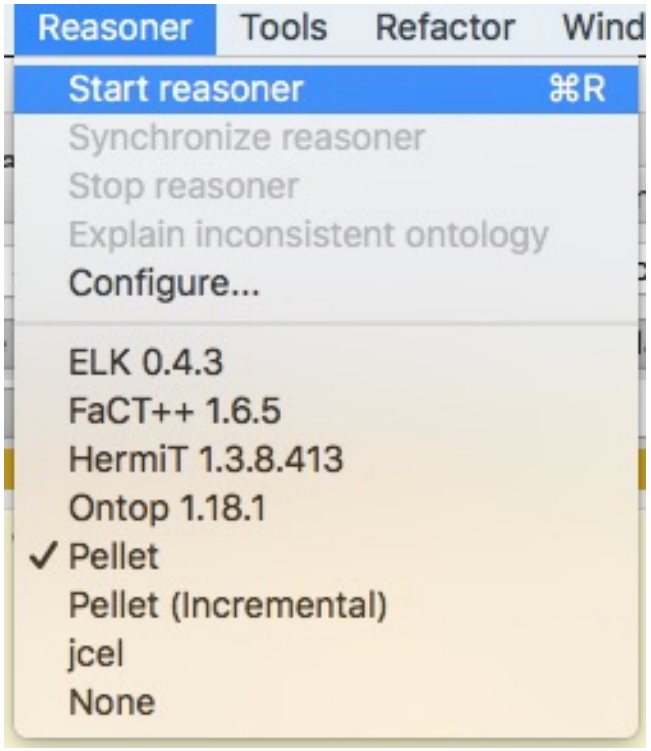
Pour créer AmericanaPizza:
- Selectionner MargheritaPizza
- Edit / Duplicate selected class
- Changer nom pour 'AmericanaPizza'
- Cliquer OK
- Ajouter axiome 'hasTopping some PepperoniSausageTopping'

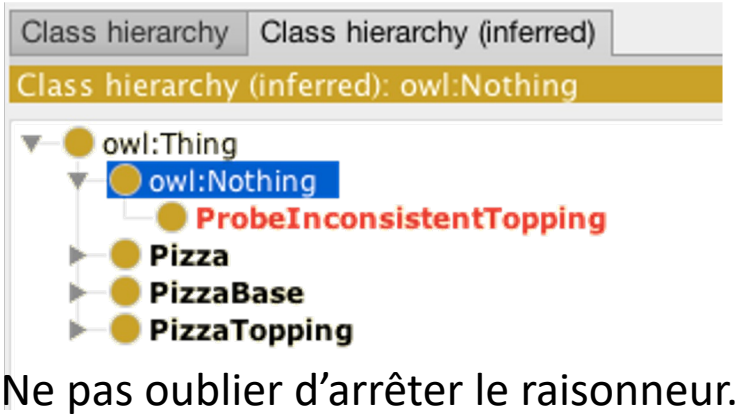
Rendre MargheritaPizza et AmericanaPizza disjointes

Remarque : Une restriction définit une classe anonyme.

Raisonneur : détecter les incohérences







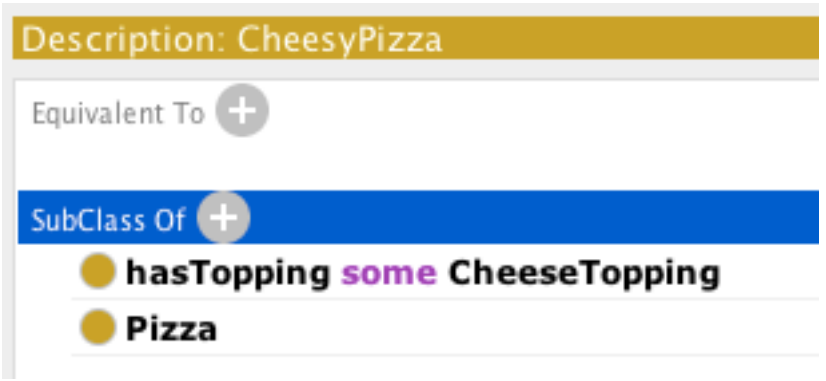
Ne pas oublier d'arrêter le raisonneur.
Réessayer en enlevant l'axiome de disjointure de CheeseTopping et VegetableTopping.

Classes primitive et classes définies (aka classes équivalentes)

Vocabulary

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

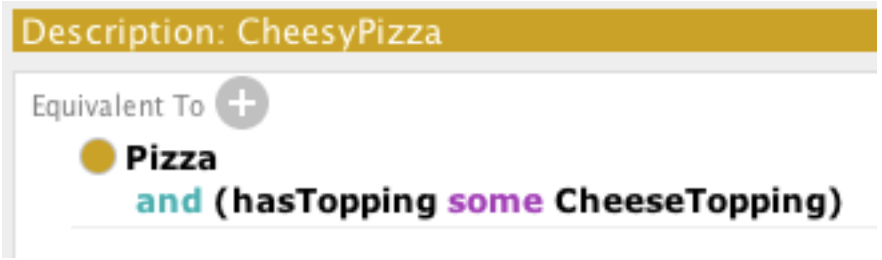
Créer CheesyPizza



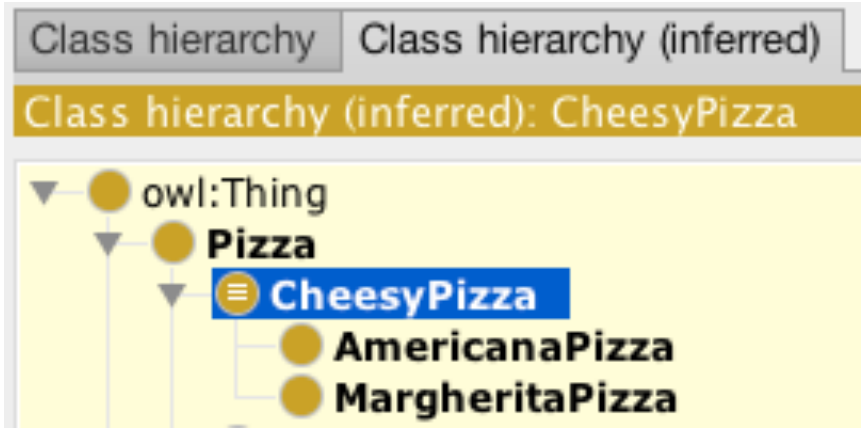
Vocabulary

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

Edit -> Convert to defined class



Raisonneur : classification automatique

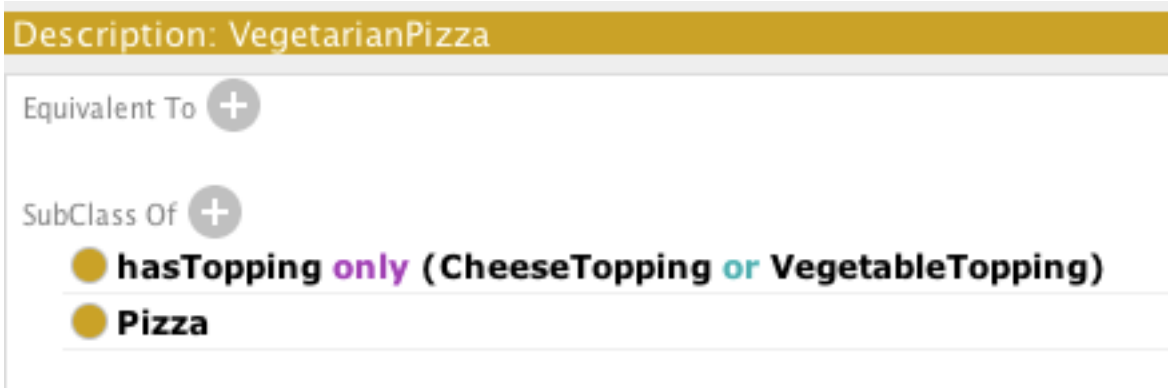


- Héritage multiple := Certaines classes ont plusieurs classes parentes.
- Principe méthodologique: Construire une hiérarchie à héritage simple asserté.
- Le raisonneur pourra inférer et maintenir l'héritage multiple.

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].

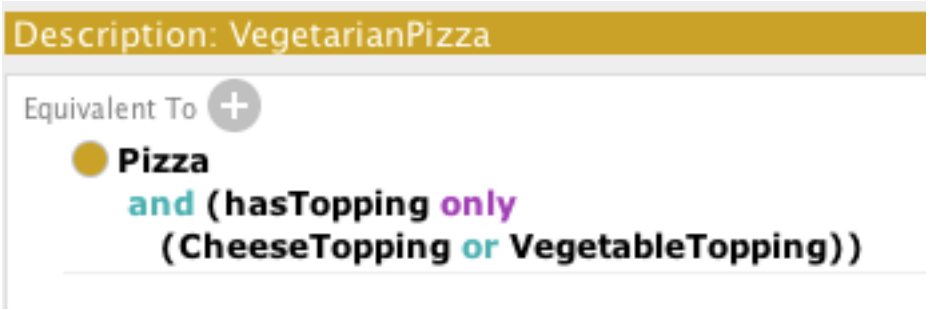
[a]The exception to this is when a property has a domain that is a primitive class. This can *coerce* classes to be reclassified under the primitive class that is the domain of the property — the use of property domains to cause such effects is strongly discouraged.

Restrictions universelles



Voyez-vous un problème avec cette caractérisation ?

Edit -> Convert to defined class



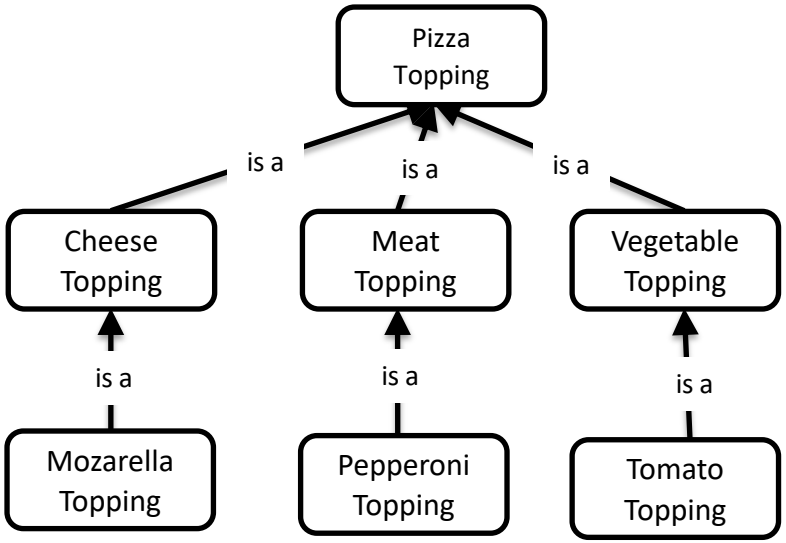
Hypothèse du monde ouvert

- MargheritaPizza sera-t-elle classée en sous-classe de VegetarianPizza ?

- *« Ce qui n'est pas énoncé peut être vrai ou faux »*

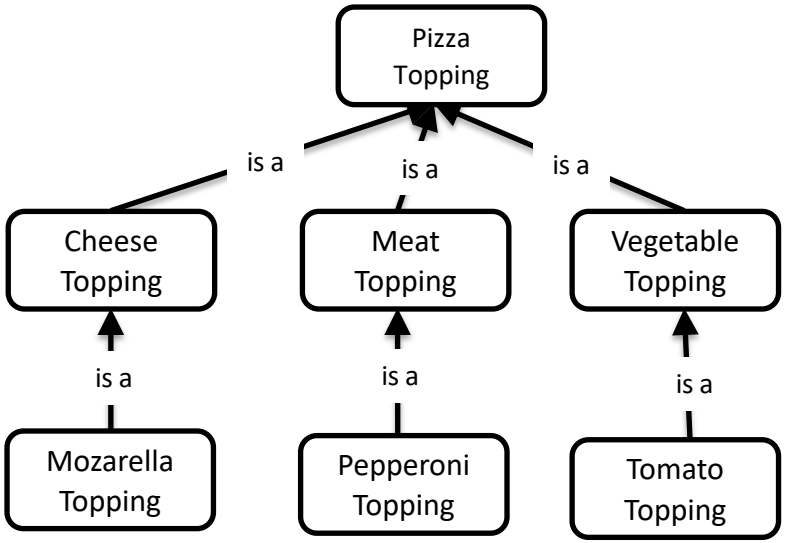
- Nécessité de contraindre l'ontologie :
  - disjonctions
  - restrictions universelles et cardinalités

Hypothèse du monde ouvert



Hypothèse du monde ouvert

BD : monde fermé



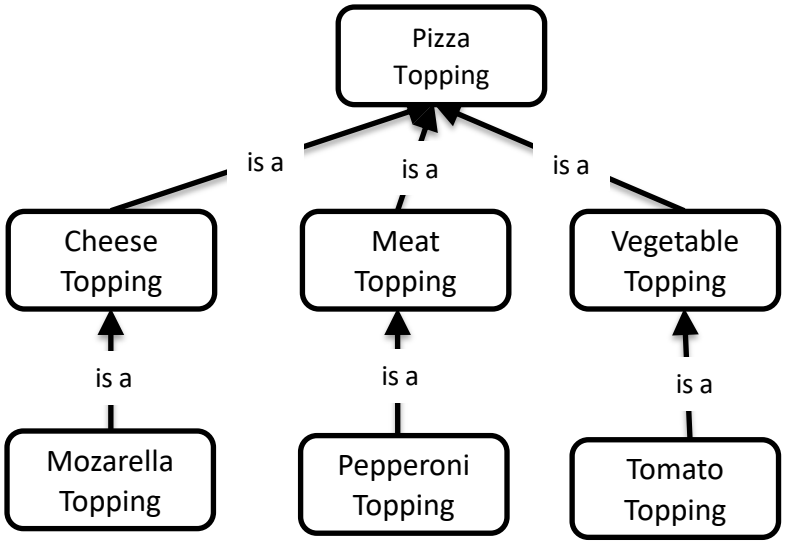
N...1

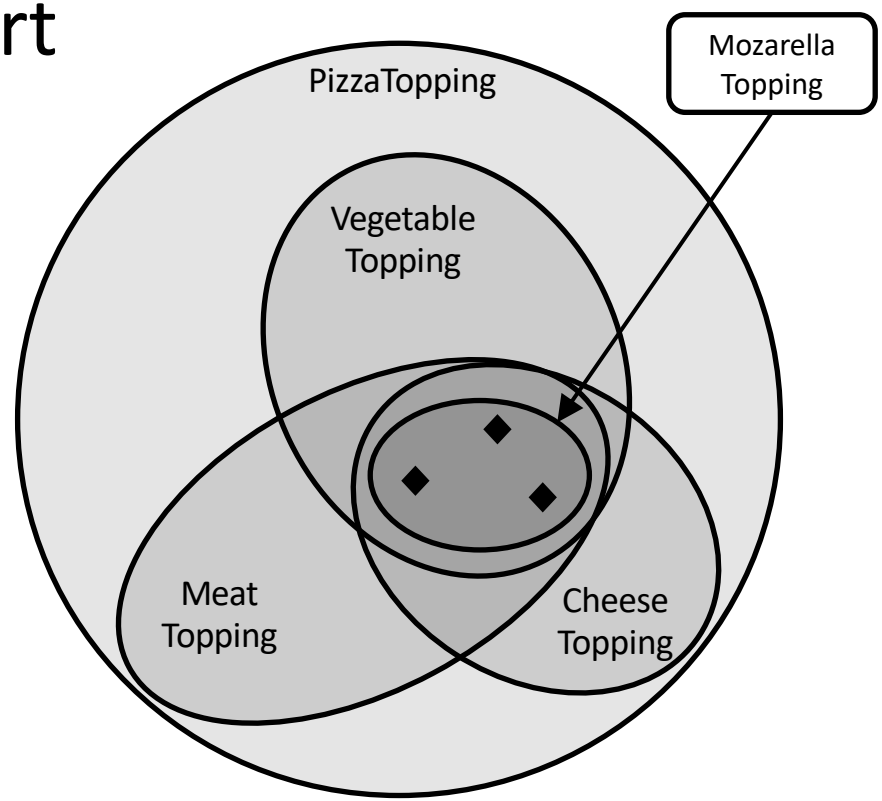
Garniture Type ID	Nom Type Garniture
T1	Fromage
T2	Viande
T3	Légumes

Garniture ID	Type Garniture	Nom Garniture
4635	T1	Mozarella
7809	T2	Pepperoni
4352	T3	Tomate

Hypothèse du monde ouvert

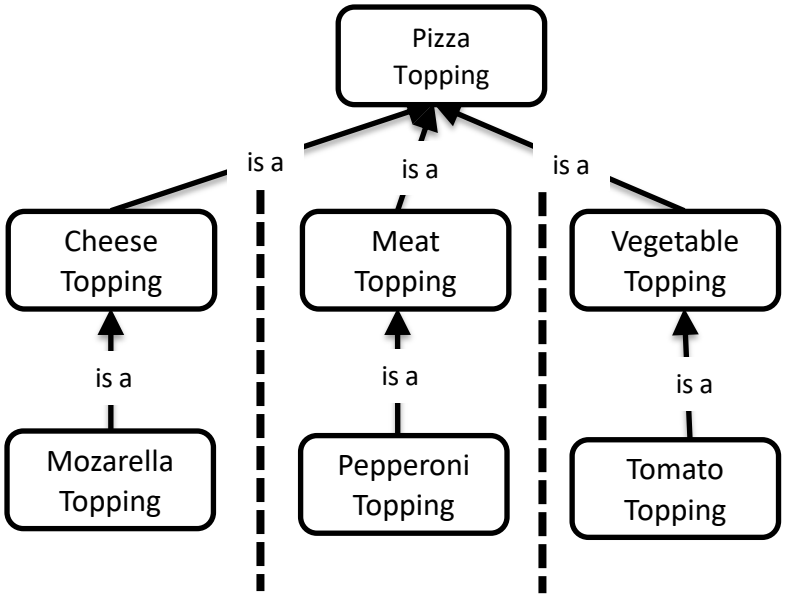
Ontologie : monde ouvert

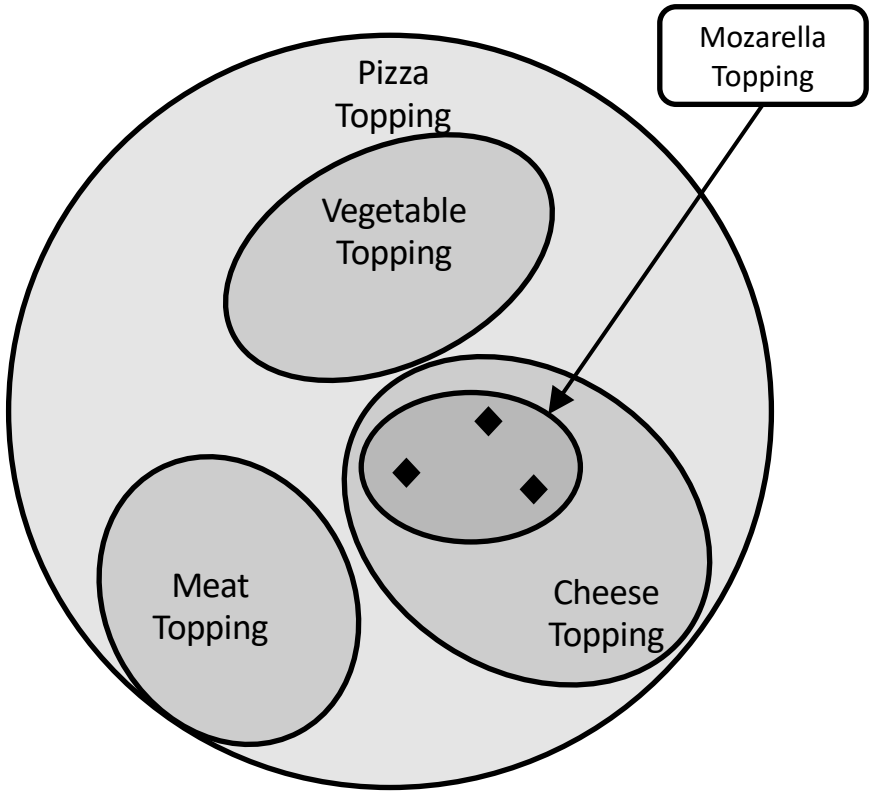




Hypothèse du monde ouvert

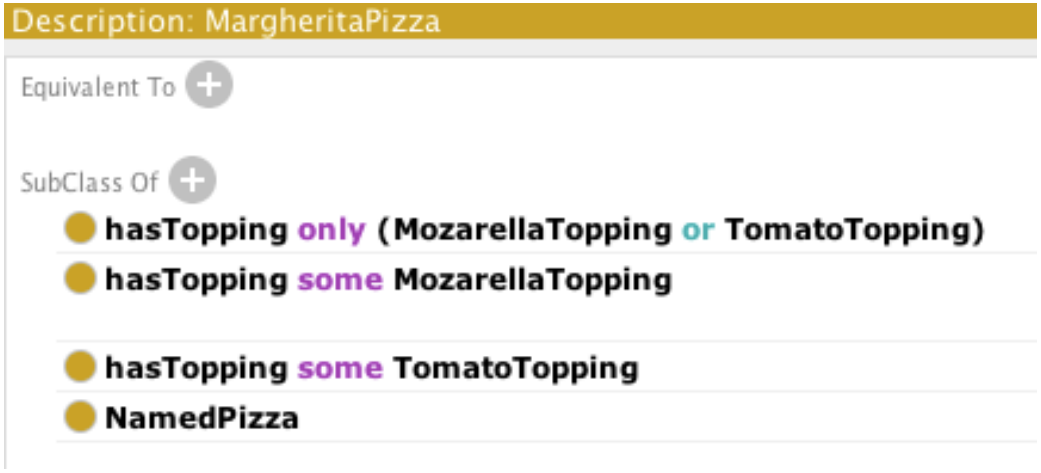
Disjonction





Raisonnement en monde ouvert

- MargheritaPizza sera-t-elle classée en sous-classe de VegetarianPizza ?
- Hypothèse de **monde ouvert** : ce qui n'est pas énoncé peut être vrai ou faux
- Pour que MargheritaPizza soit classée en sous-classe de VegetarianPizza, il faut ajouter un axiome de fermeture.

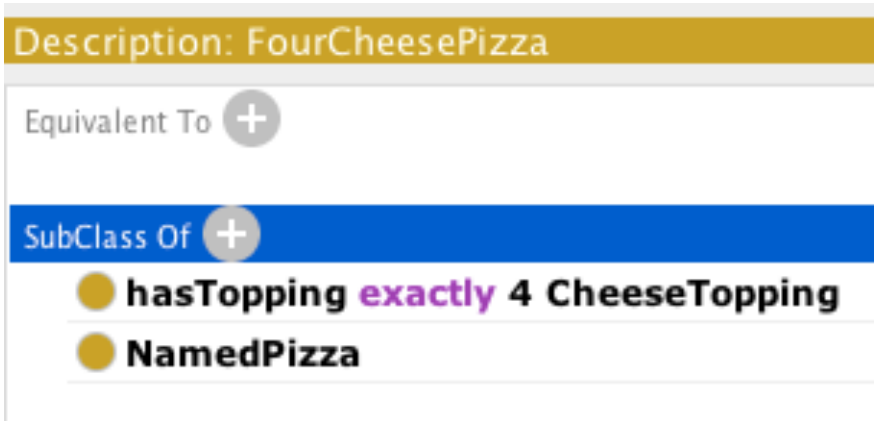


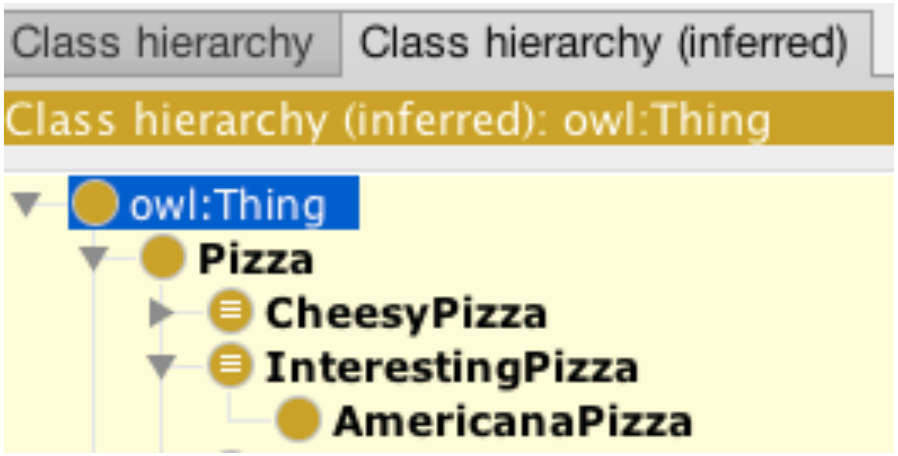
Clic droit sur axiome existentiel -> 'Create closure axiom'

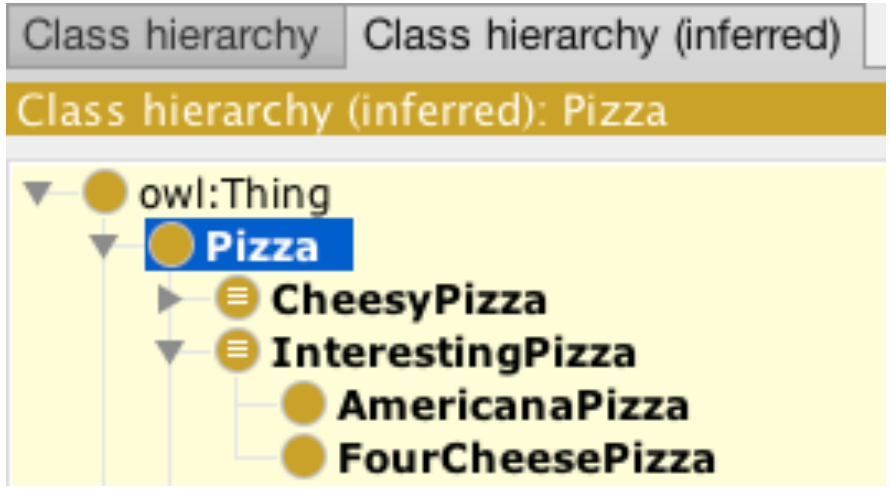
Remarque : Les axiomes existentiels sont également importants, pour éviter qu'une pizza sans garniture puisse être considérée comme MargheritaPizza.

Restrictions de cardinalité

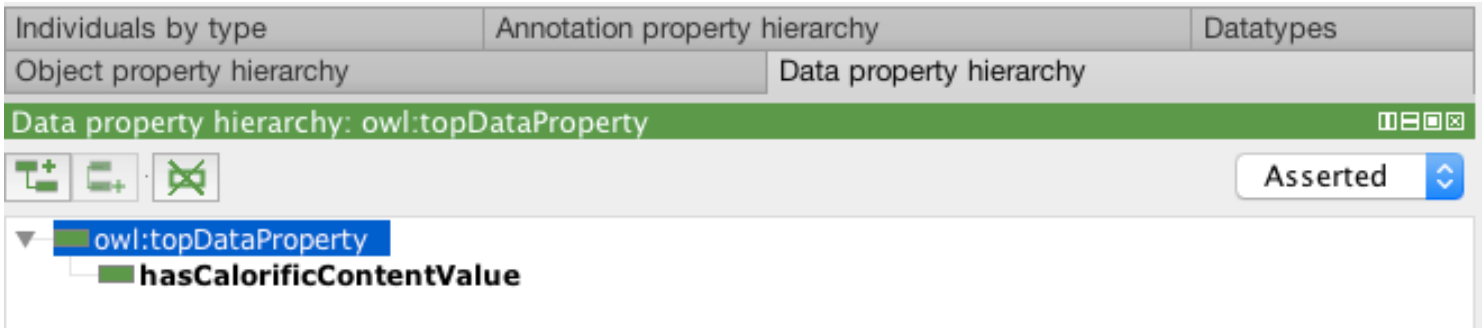
Créer InterestingPizza sous-classe de Pizza

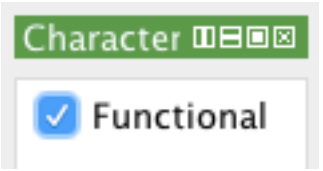


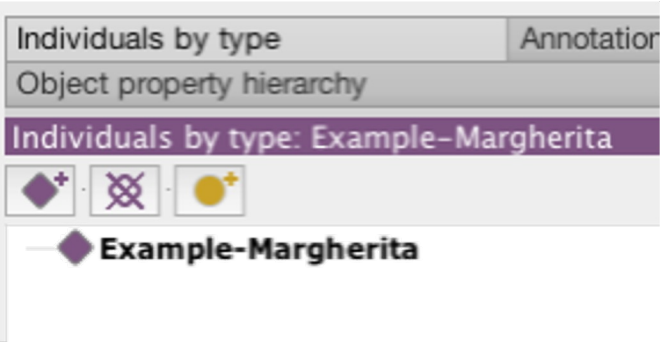


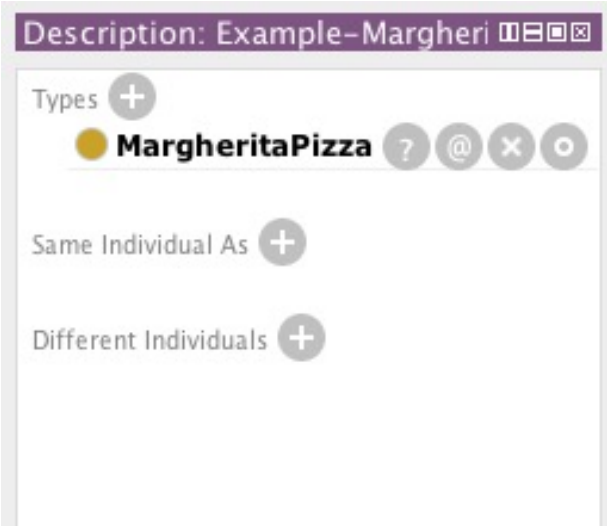


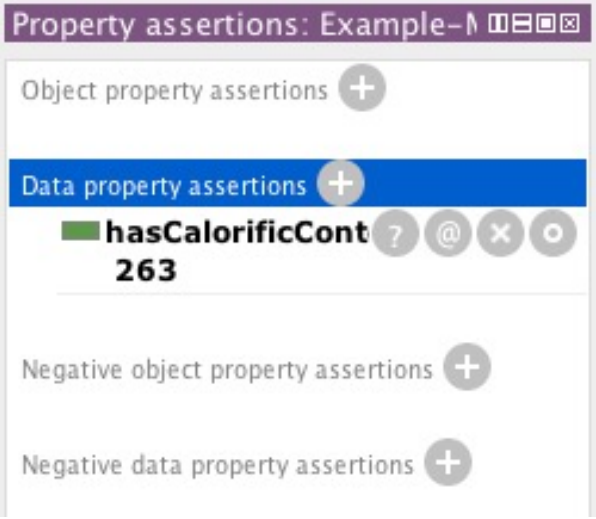
Datatype properties et individus

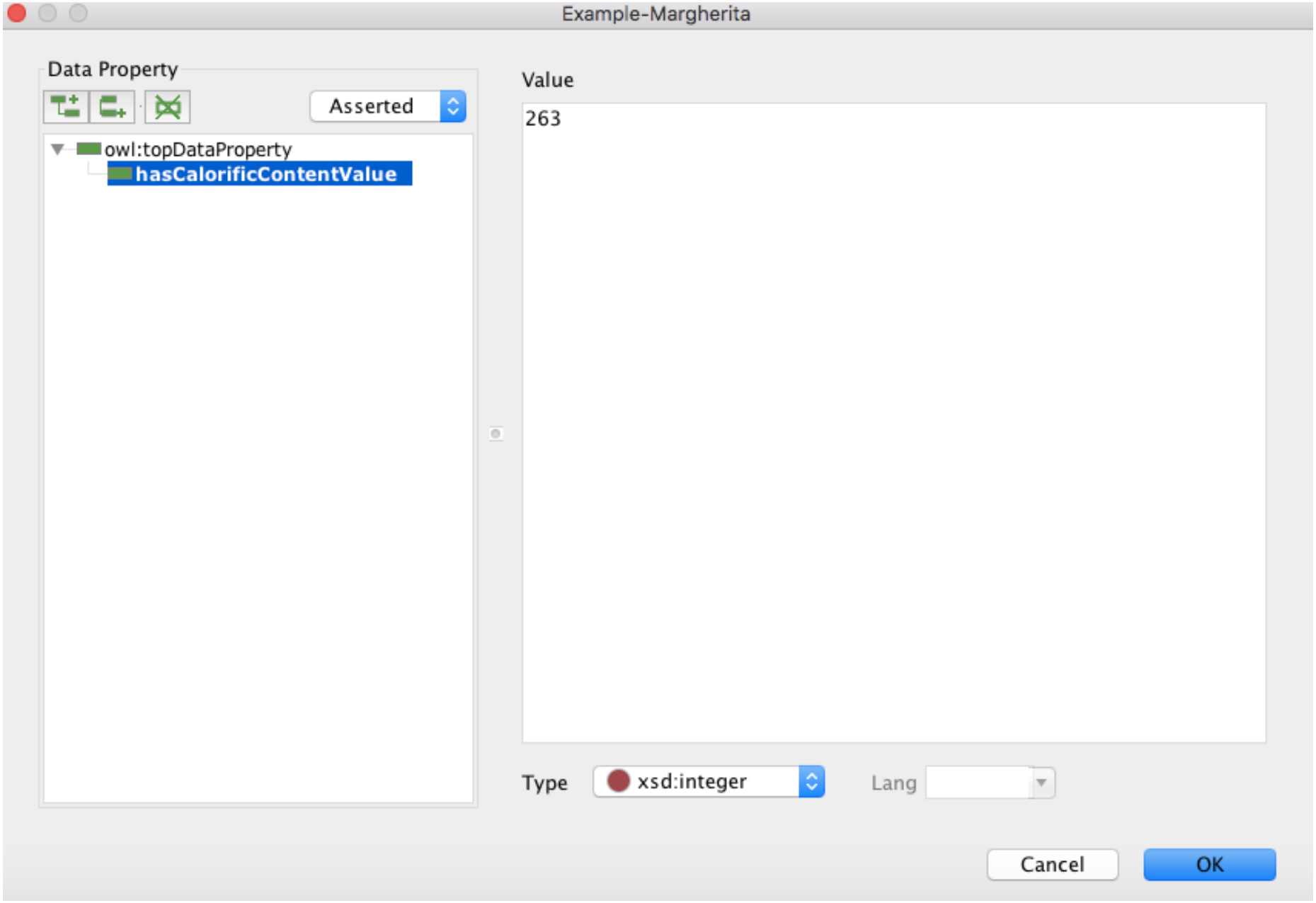




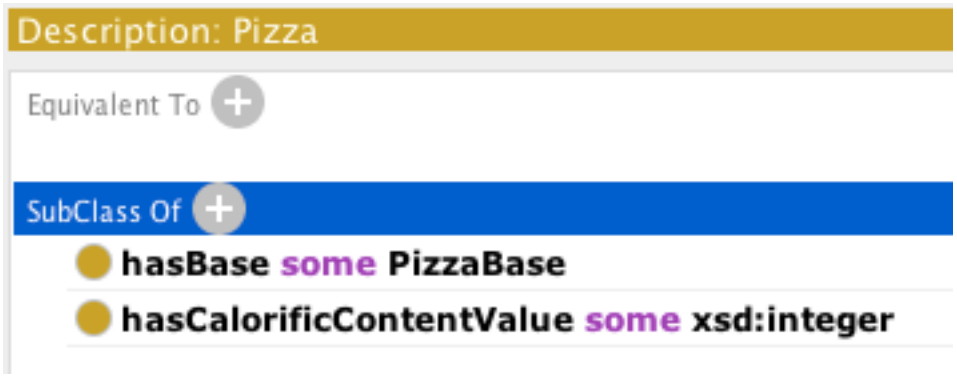


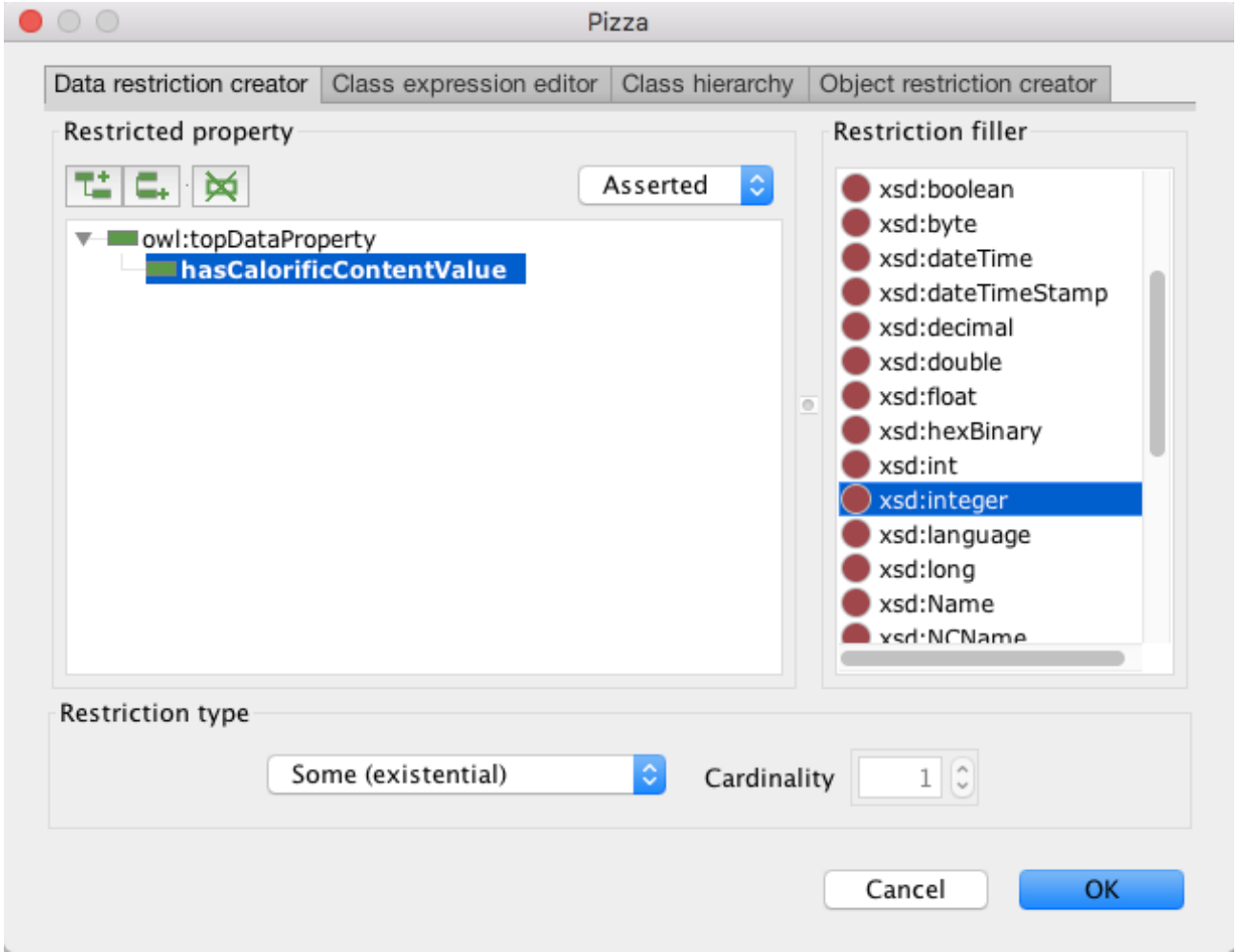




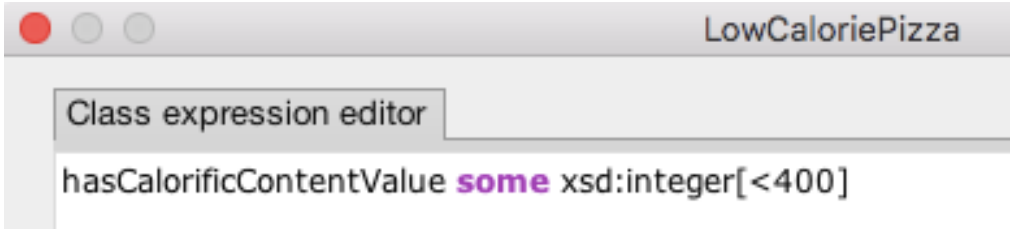

Example-Margherita
Data Property
Asserted
owl:topDataProperty
hasCalorificContentValue
Value
263
Type
xsd:integer
Lang
Cancel
OK

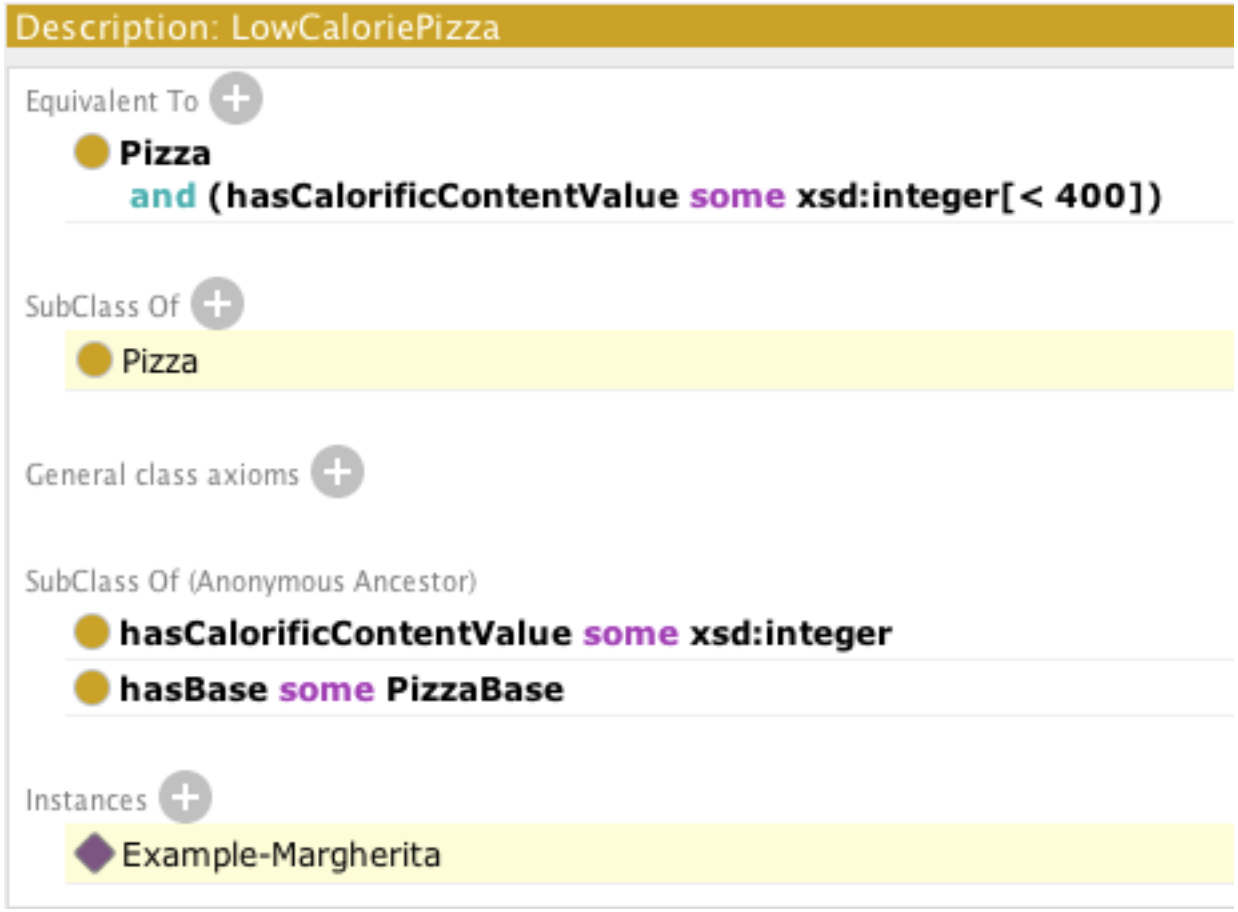
Axiome : Toutes les pizzas ont une valeur calorifique





Raisonner avec des datatype properties





Analyse ontologique des pays

- On voudrait pouvoir dire qu'une garniture mozarella a comme pays d'origine l'Italie.

- Les pays sont-ils des classes ou des individus?

Relation entre une classe et un individu