

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 Fabry²

¹ CNRS, IRIT

² GRIIS, Université de Sherbrooke

4 juin 2024



GRIIS

Protégé : un outil de création d'ontologies

The screenshot shows the Protégé ontology editor interface with several tabs open:

- Class hierarchy:** Shows the class hierarchy for 'drug administration specification'. The tree includes nodes like 'information content entity', 'condition', 'data item', 'datum label', 'directive information entity', 'dose specification', 'normative specification', 'drug administration specification' (which is 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', and 'site of administration specification'.
- Annotations:** Displays annotations for 'drug administration specification'. 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 :**
- Description:** Shows the description of 'drug administration specification'. It lists:
 - 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**: +
- Object property hierarchy:** Shows the object property hierarchy. The top node is 'topObjectProperty'.
- Data property hierarchy:** Shows the data property hierarchy.
- Individuals by type:** Shows individuals categorized by type.
- Annotation property hierarchy:** Shows the annotation property hierarchy.
- Datatypes:** Shows the datatypes.

At the bottom, there are buttons for "To use the reasoner click Reasoner->Start reasoner" and "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

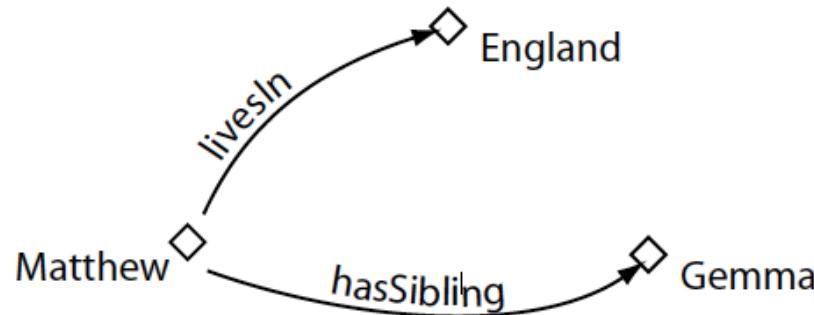


Figure 3.2: Representation Of Properties

Classes

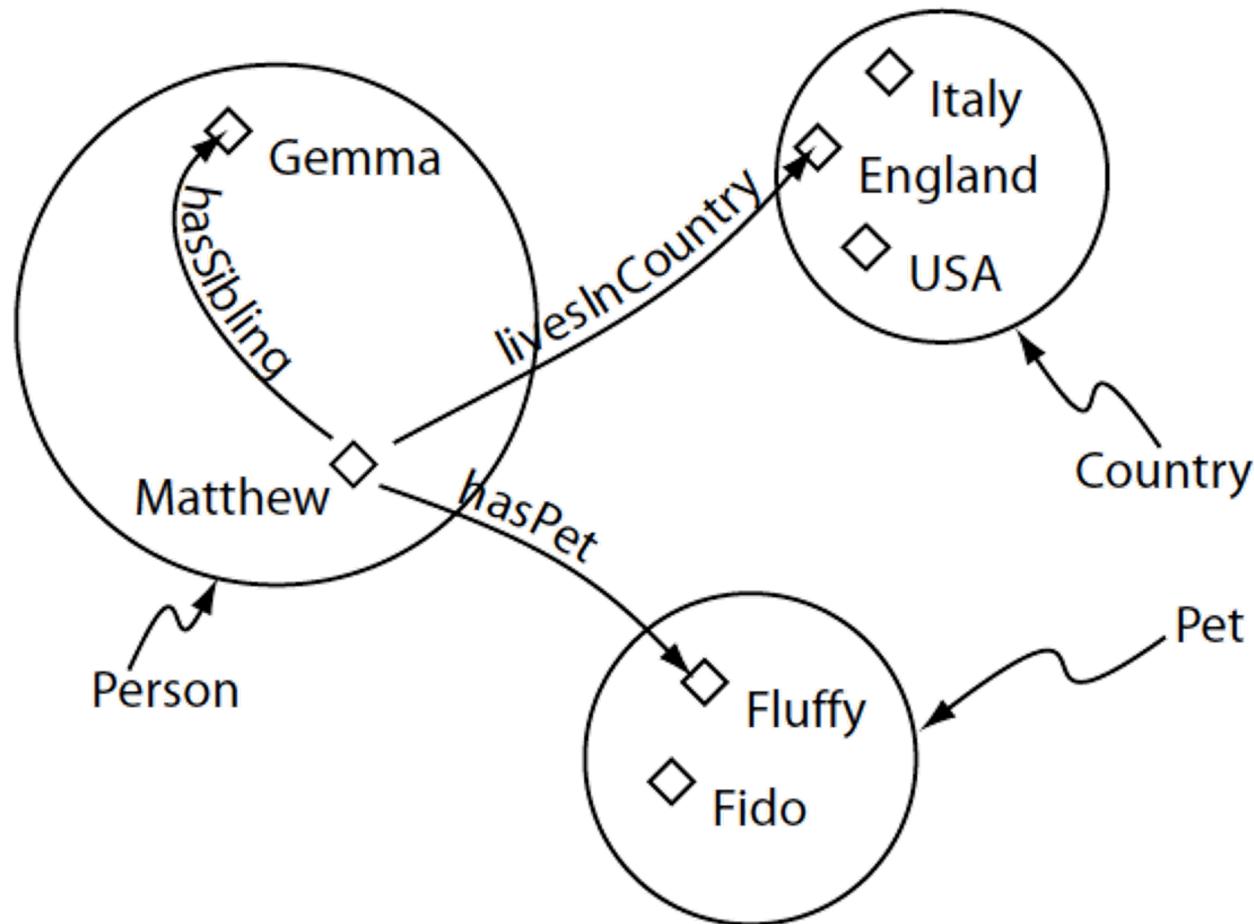
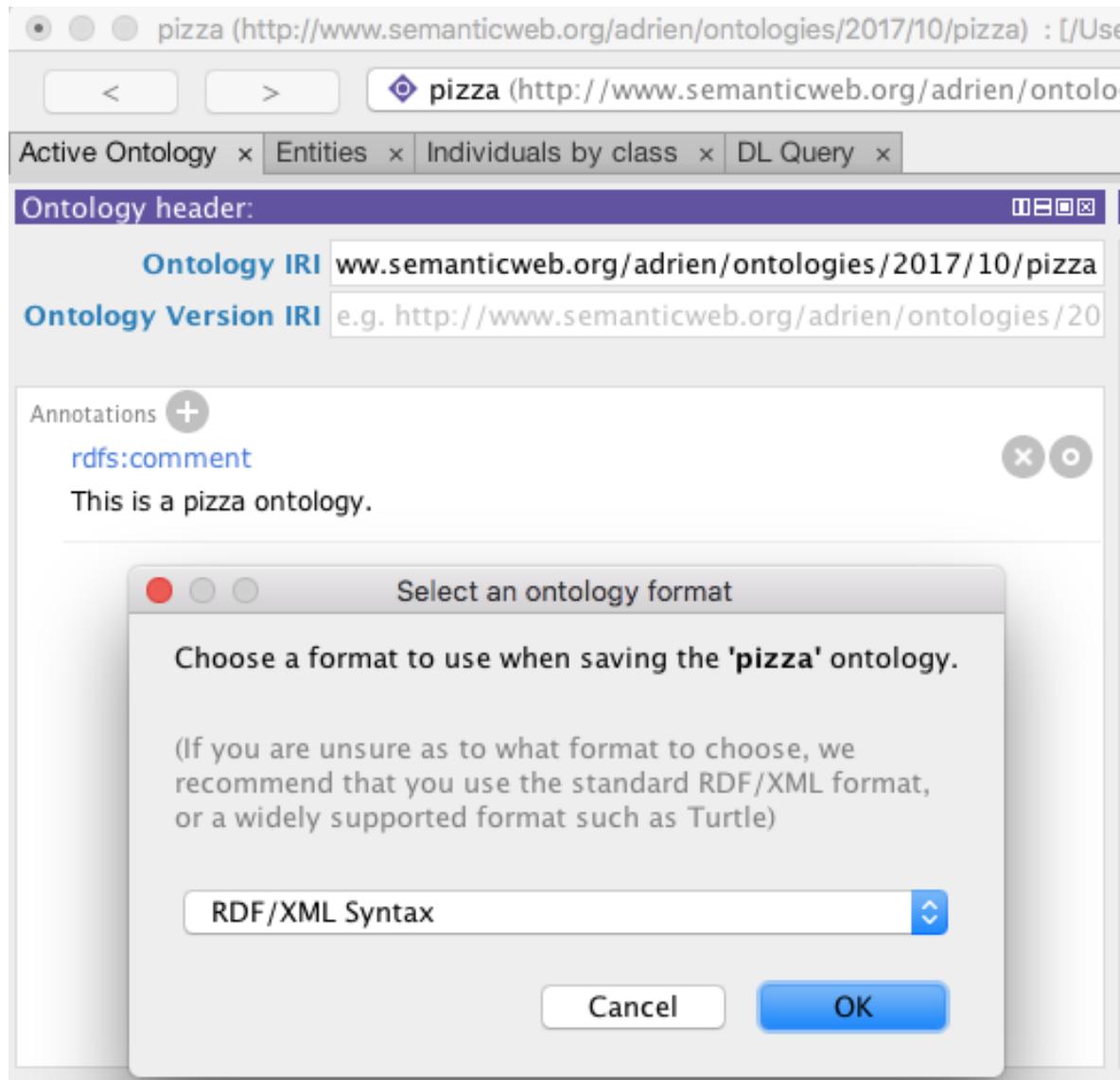
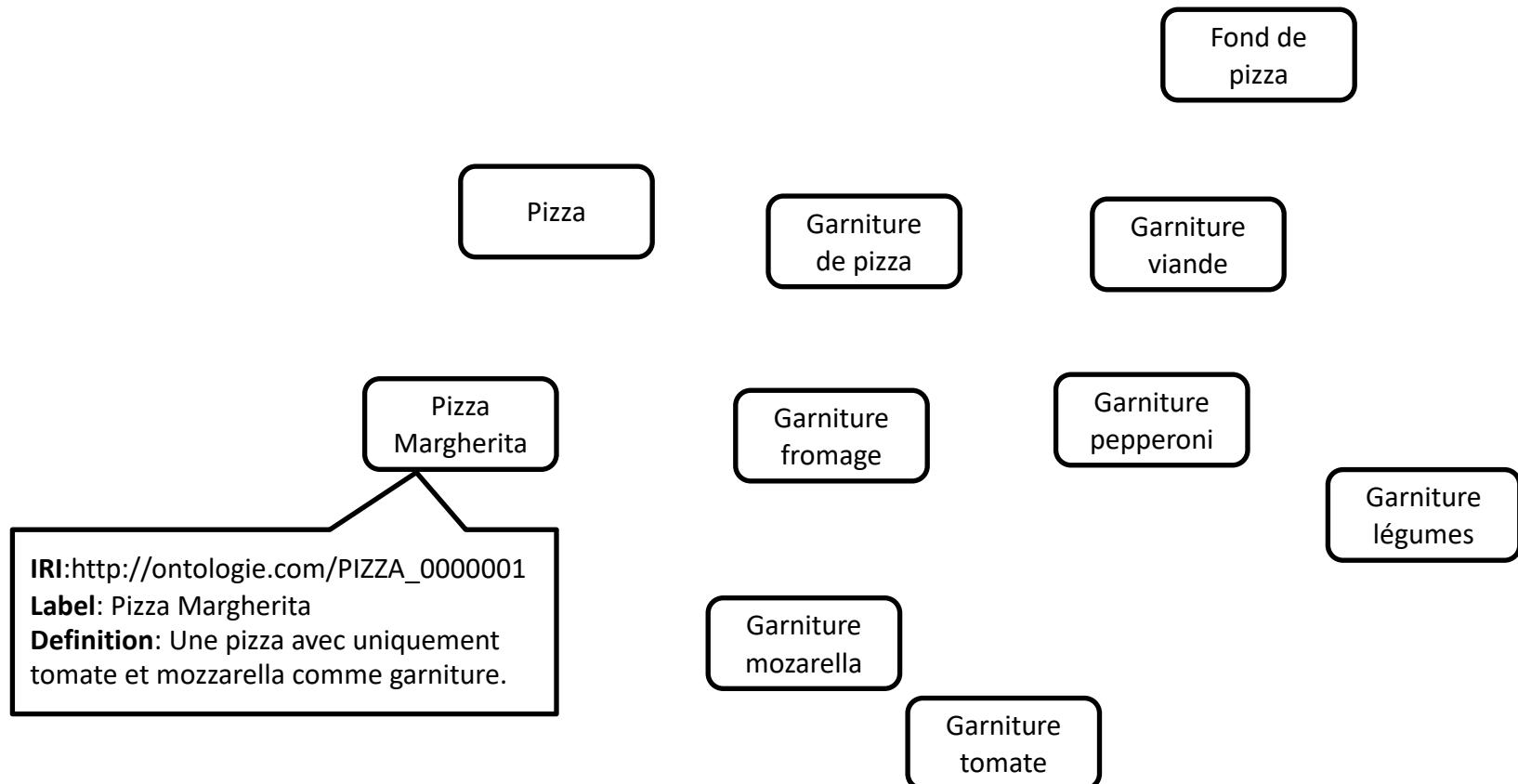


Figure 3.3: Representation Of Classes (Containing Individuals)

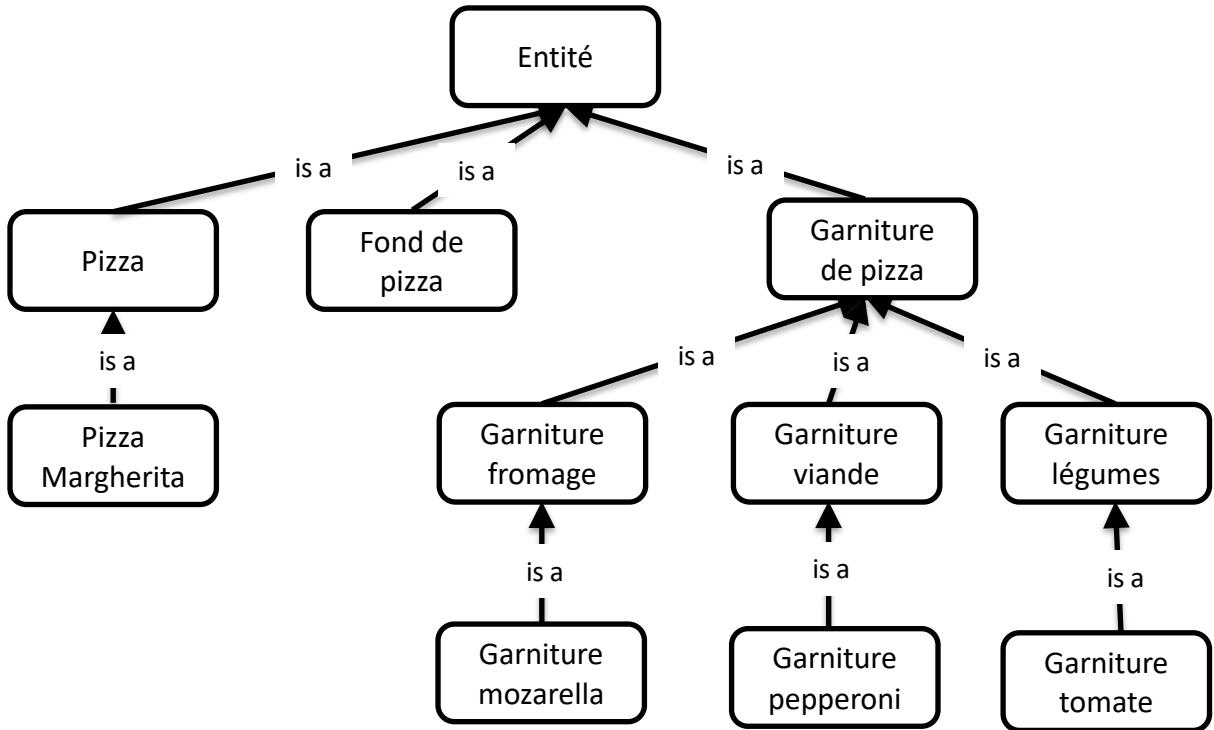
Créer une ontologie sous Protégé 5.x



Quelles classes pour une ontologie des pizzas ?



Une première taxonomie des pizzas



Une première taxonomie des pizzas

a comme
fond

Entité

```
<!-- 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

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

Class hierarchy Class hierarchy (inferred)

Class hierarchy: Pizza

owl:Thing
PizzaTopping
PizzaBase
Pizza

Description: Pizza

Equivalent To +

SubClass Of +

General class axioms +

SubClass Of (Anonymous Ancestor)

Instances +

Target for Key +

Disjoint With +

Class hierarchy Expression editor

owl:Thing
Pizza
PizzaBase
PizzaTopping

10

Créer des taxonomies de classes

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

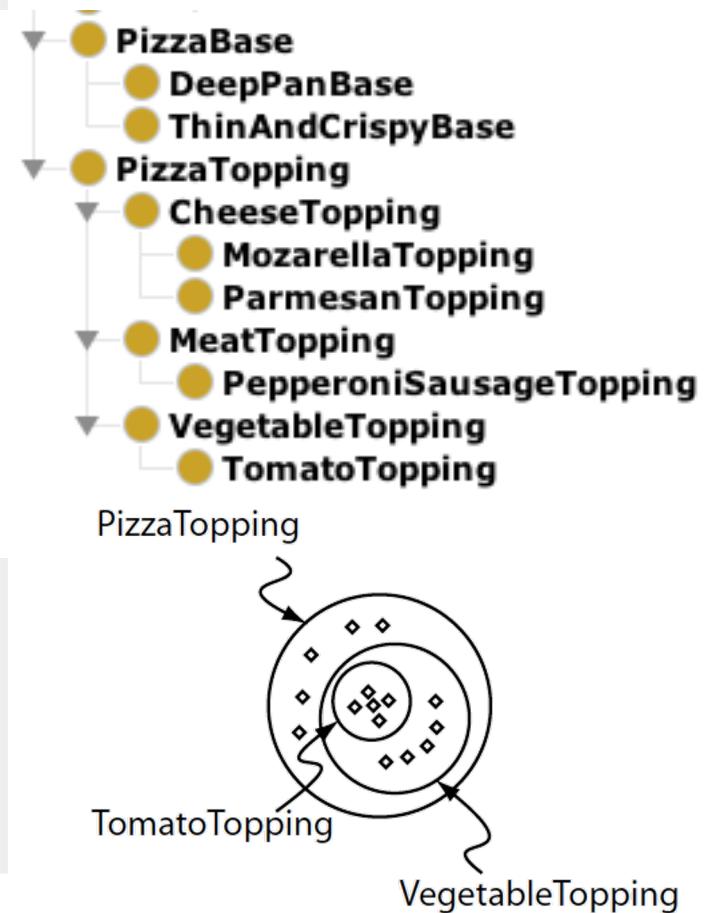
Enter hierarchy

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

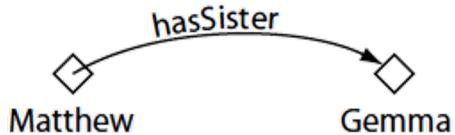
```
Cheese
    Mozarella
    Parmesan
Meat
    PepperoniSausage
Vegetable
    Tomato
```

Prefix

Suffix Topping



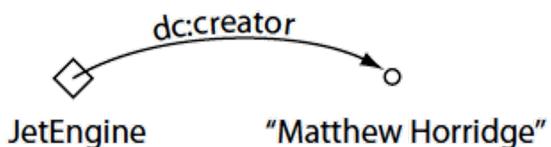
Trois types de propriétés (aka relations)



An object property linking the individual Matthew to the individual Gemma



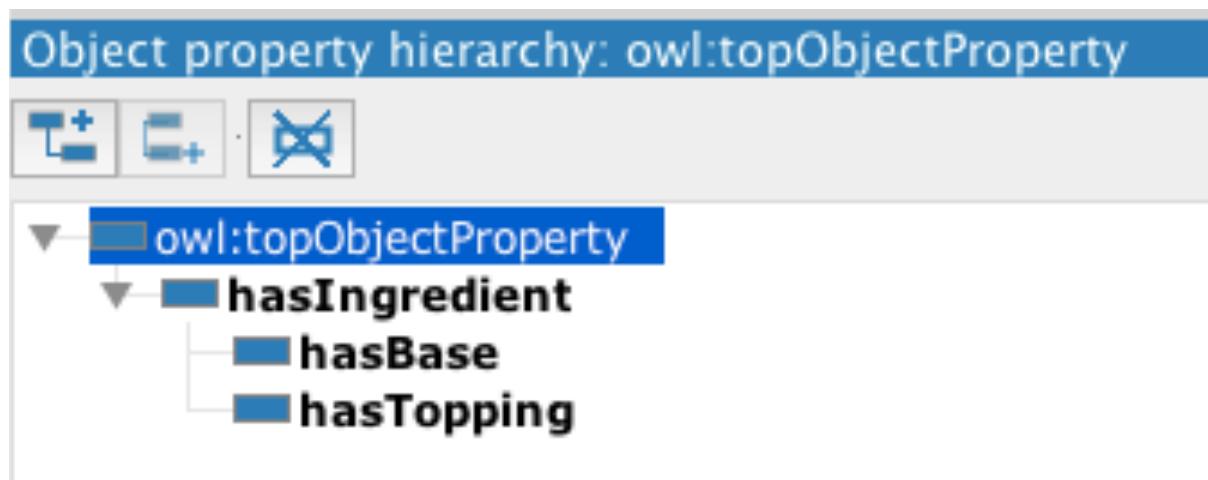
A datatype property linking the individual Matthew to the data literal '25', which has a type of an xsd:integer.



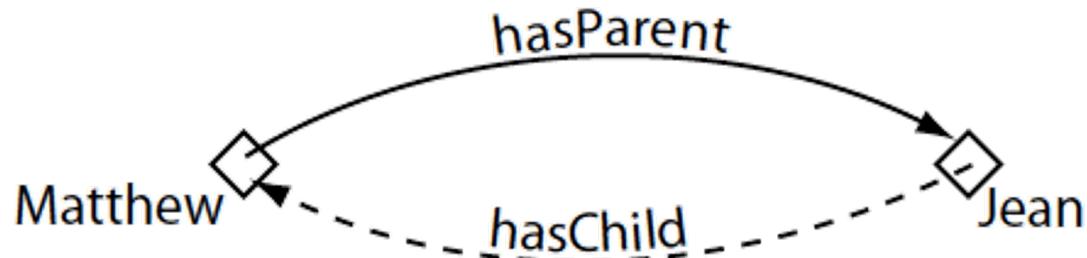
An annotation property, linking the class 'JetEngine' to the data literal (string) "Matthew Horridge".

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

hasIngredient

isIngredientOf

Cancel OK

Description: hasIngredient

Equivalent To +

SubProperty Of +

Inverse Of +

Domains (intersection) +

Ranges (intersection) +

Disjoint With +

SuperProperty Of (Chain) +

The central part of the image shows a dialog box titled "hasIngredient" with a tree view of object properties. The tree shows "owl:topObjectProperty" as the root, with "hasIngredient" and "isIngredientOf" as children. The "isIngredientOf" node is highlighted in blue. At the bottom of the dialog are "Cancel" and "OK" buttons.

Caractéristique des relations

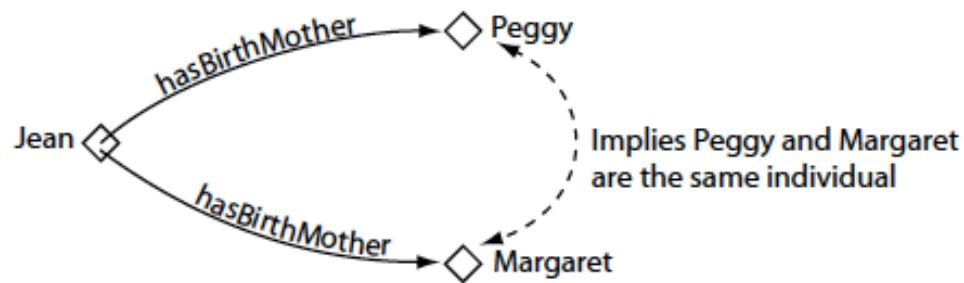


Figure 4.18: An Example Of A Functional Property: `hasBirthMother`

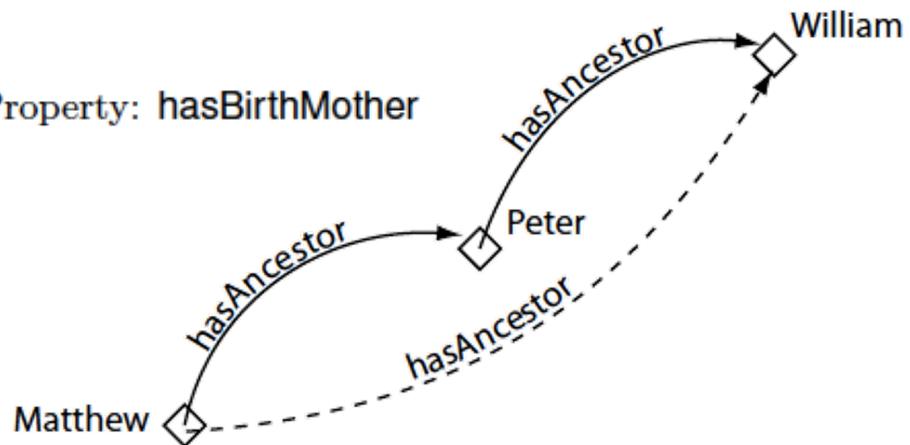


Figure 4.20: An Example Of A Transitive Property: `hasAncestor`

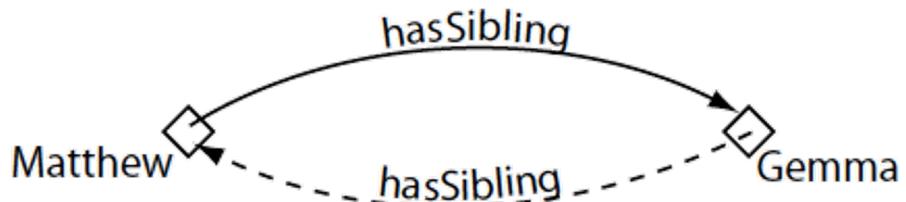


Figure 4.21: An Example Of A Symmetric Property: `hasSibling`

Caractéristiques des relations : application

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

The screenshot shows a semantic web editor interface with several tabs at the top: 'Individuals by type', 'Annotation property hierarchy', 'Datatypes', 'Object property hierarchy', and 'Data property hierarchy'. The 'Object property hierarchy' tab is active, showing a tree structure under 'Object property hierarchy: hasIngredient'. The tree includes 'owl:topObjectProperty', 'isIngredientOf', 'isToppingOf', 'isBaseOf', 'hasIngredient' (which is selected), 'hasBase', and 'hasTopping'. On the right side of the editor, there is a panel titled 'Characteristics: hasIngredient' with the following options:

Description	Characteristics
Characteristics: hasIngredient	<input checked="" type="checkbox"/> Functional <input type="checkbox"/> Inverse functional <input checked="" type="checkbox"/> Transitive <input type="checkbox"/> Symmetric <input type="checkbox"/> Asymmetric <input type="checkbox"/> Reflexive <input type="checkbox"/> Irreflexive

Domain & Range

The screenshot shows the Protégé ontology editor. On the left, the 'Object property hierarchy' for `hasTopping` is displayed. The tree structure includes `owl:topObjectProperty`, `isIngredientOf`, `isToppingOf`, `isBaseOf`, `hasIngredient`, `hasBase`, and `hasTopping`. On the right, the asserted range for `hasTopping` is shown as `Pizza`. Below it, the ranges intersection is shown as `PizzaTopping`, and there is a 'Disjoint With' section.

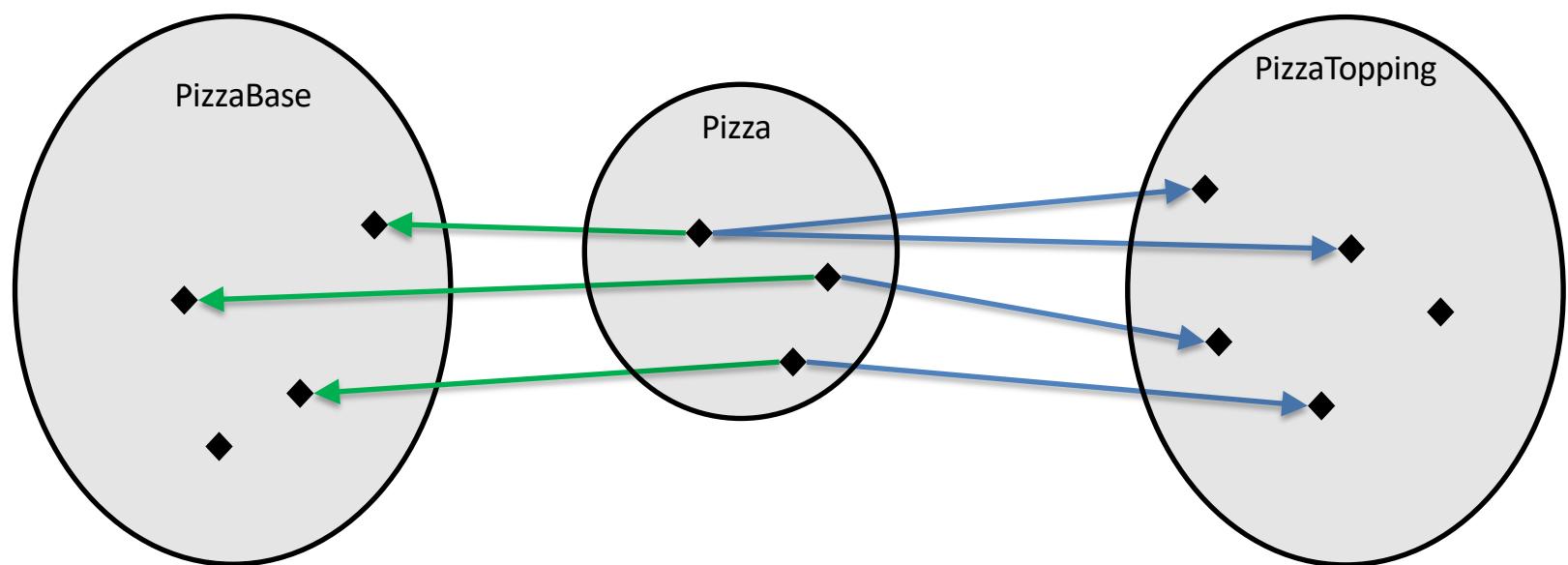
`hasBase`: Domain Pizza, Range PizzaBase (utiliser autocomplete)



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`

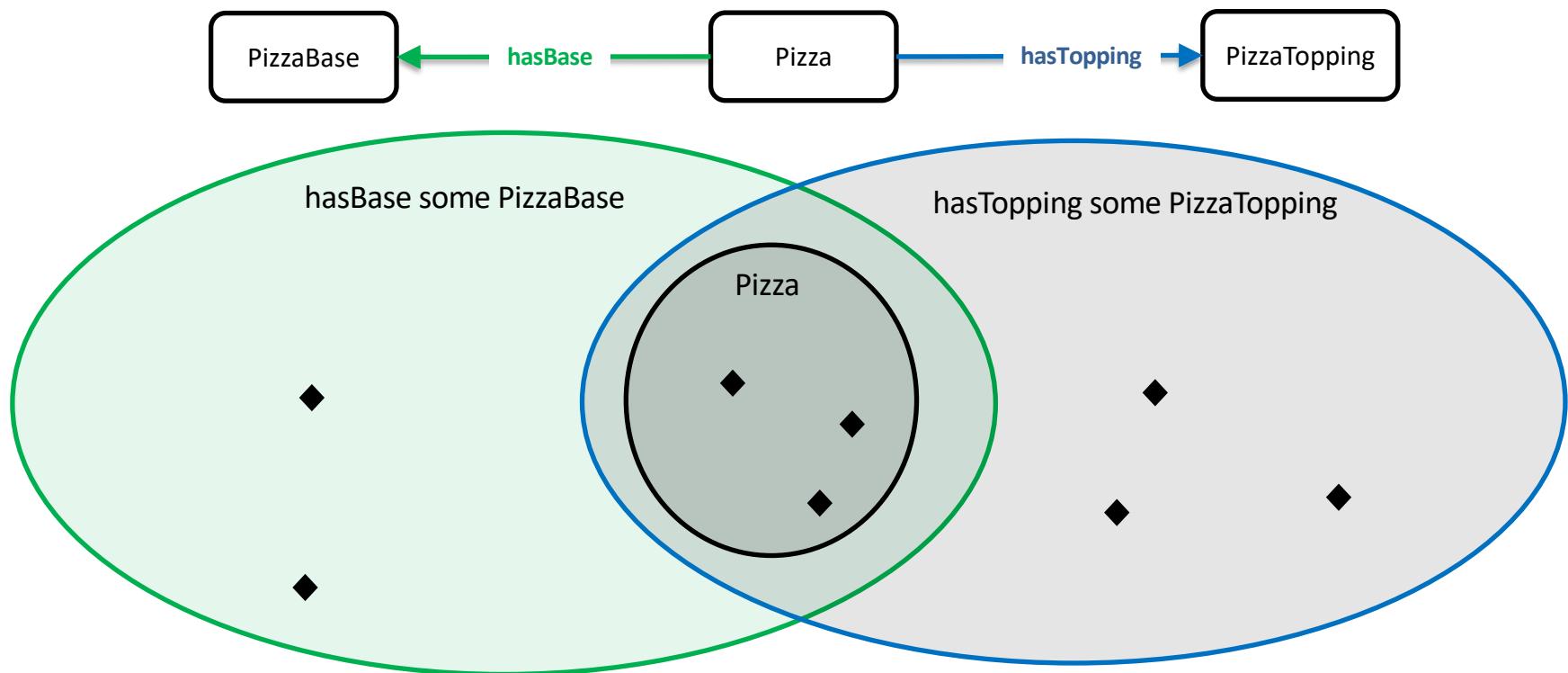
Axiomes existentiels (restrictions existentielles)



Axiomes existentiels (restrictions existentielles)

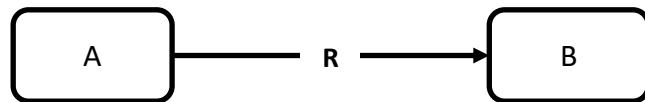


A SubclassOf R some B



Axiomes existentiels (restrictions existentielles)

Un axiome est une restriction de classe



A SubClassOf (R *some* B)

SubClass Of +
• R some B

Axiomes existentiels (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

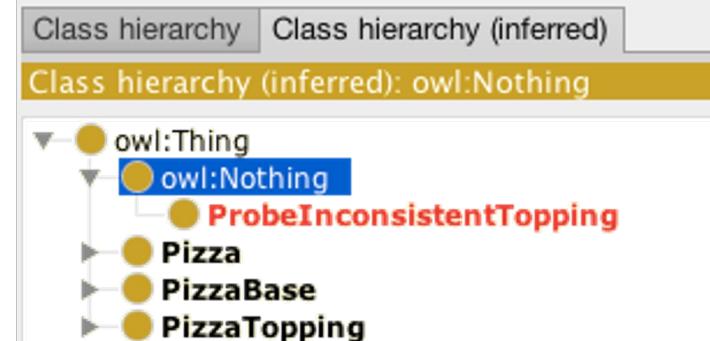
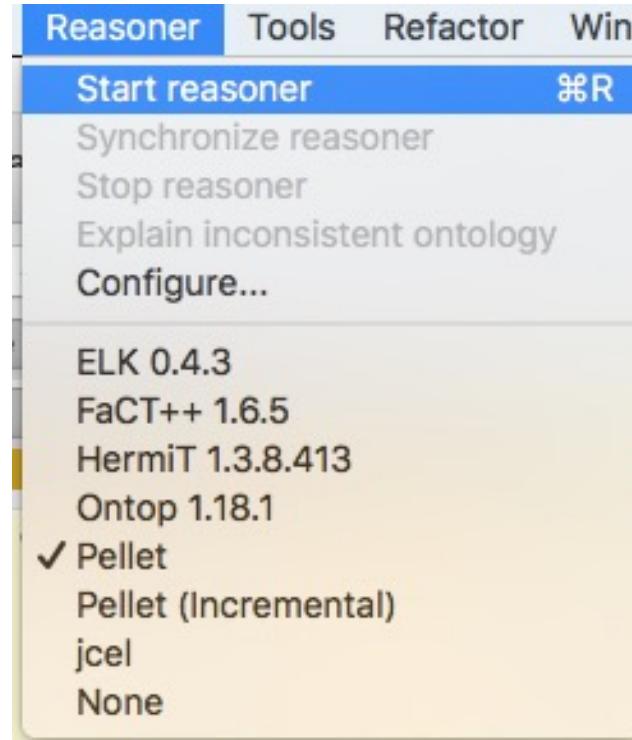
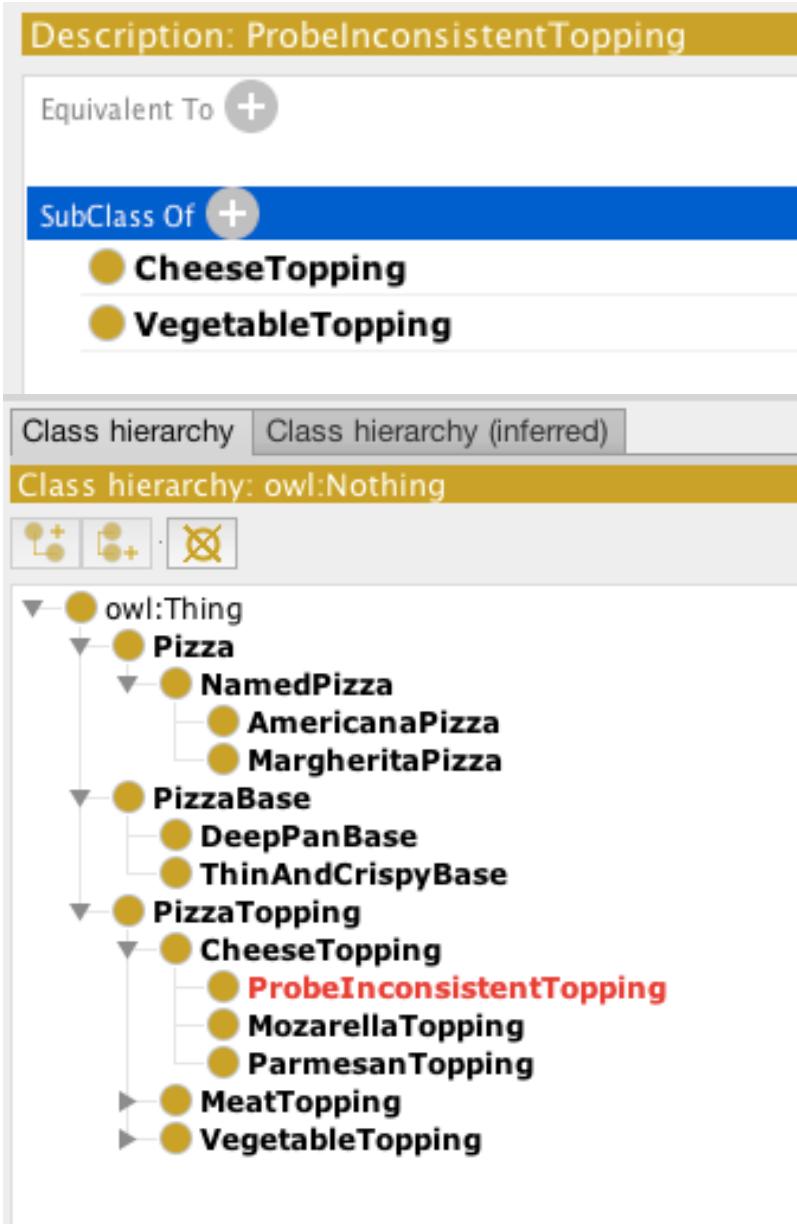
Pour créer AmericanaPizza:

- 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

Description: CheesyPizza

Equivalent To +

SubClass Of +

- hasTopping some CheeseTopping
- Pizza

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

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 subclasses 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

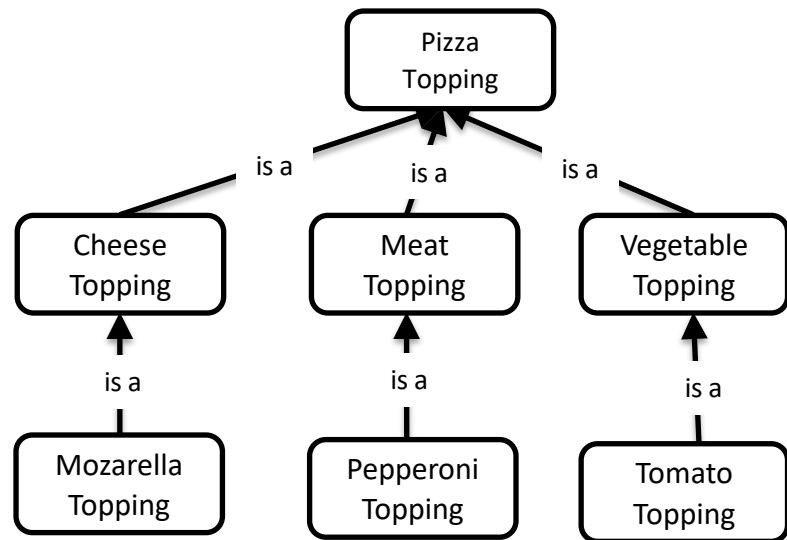
Equivalent To +

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

Hypothèse du monde ouvert

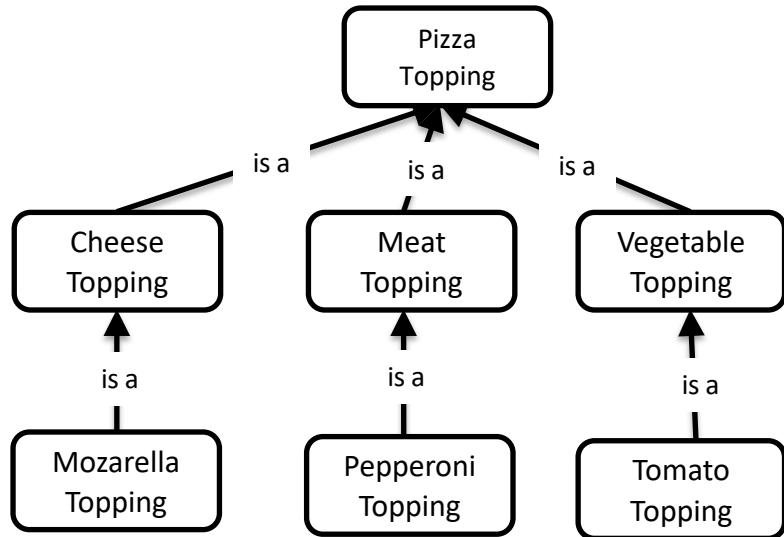
- MargheritaPizza sera-t-elle classée en sous-classe de VegetarianPizza ?
- « *Ce qui n'est pas énoncé peut être vrai ou faux* »
- Nécessité de contraindre l'ontologie :
 - disjonctions
 - restrictions universelles et cardinalités

Hypothèse du monde ouvert



Hypothèse du monde ouvert

BD : monde fermé



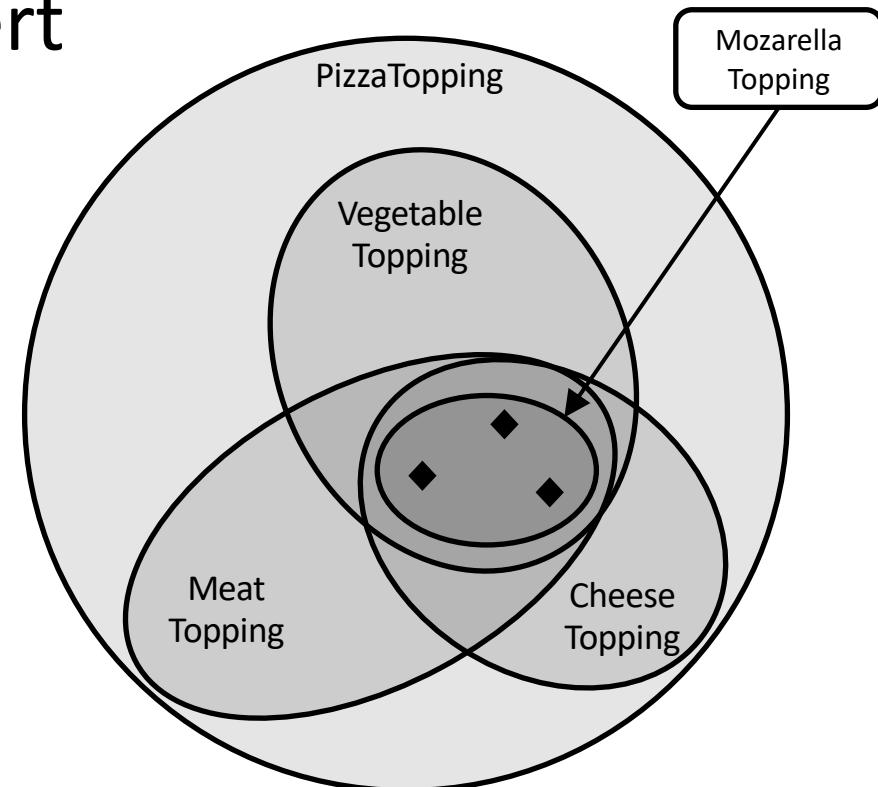
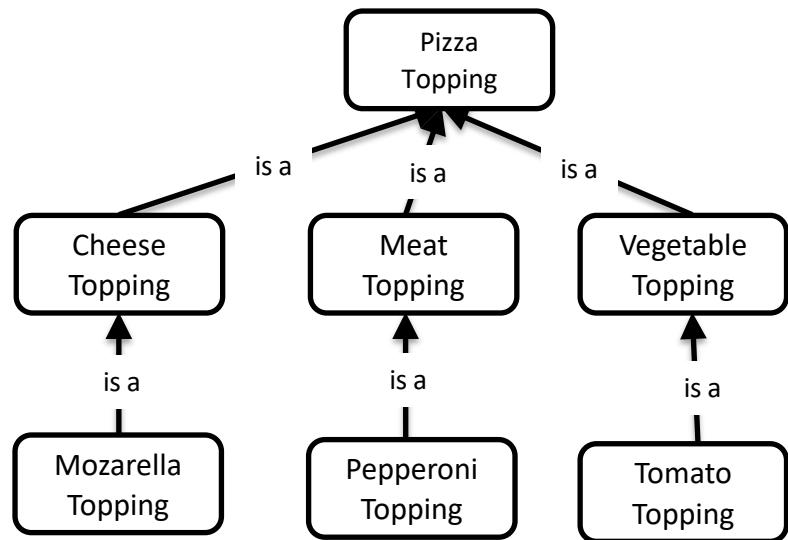
Garniture Type ID	Nom Type Garniture
T1	Fromage
T2	Viande
T3	Légumes

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

The diagram shows two tables. The top table maps generic topping types to names. The bottom table provides specific instances of these types, each associated with a unique ID.

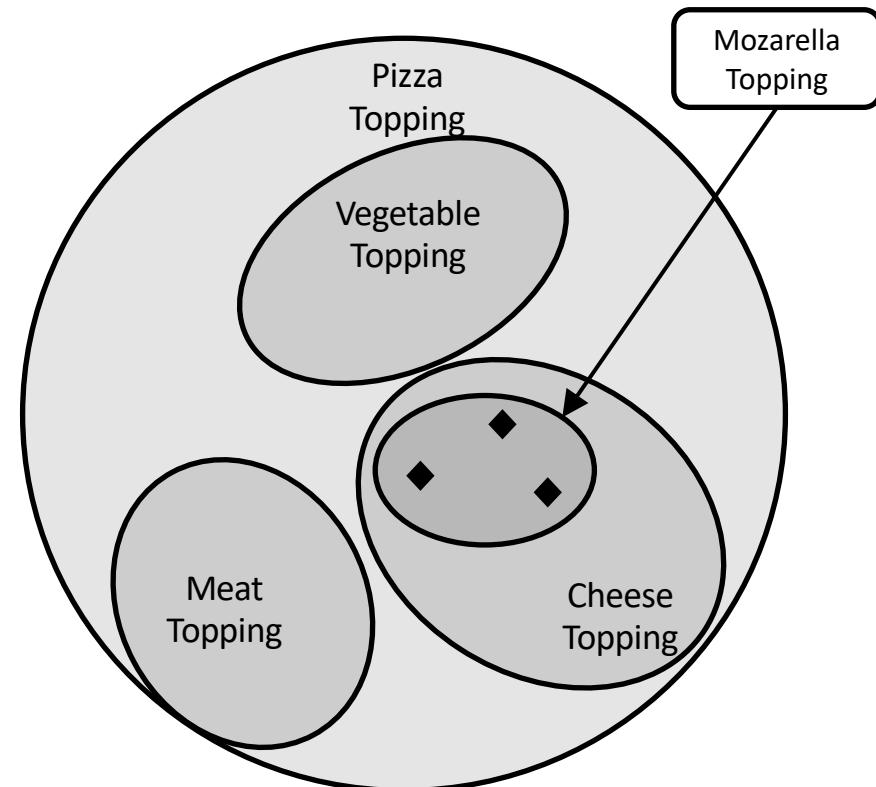
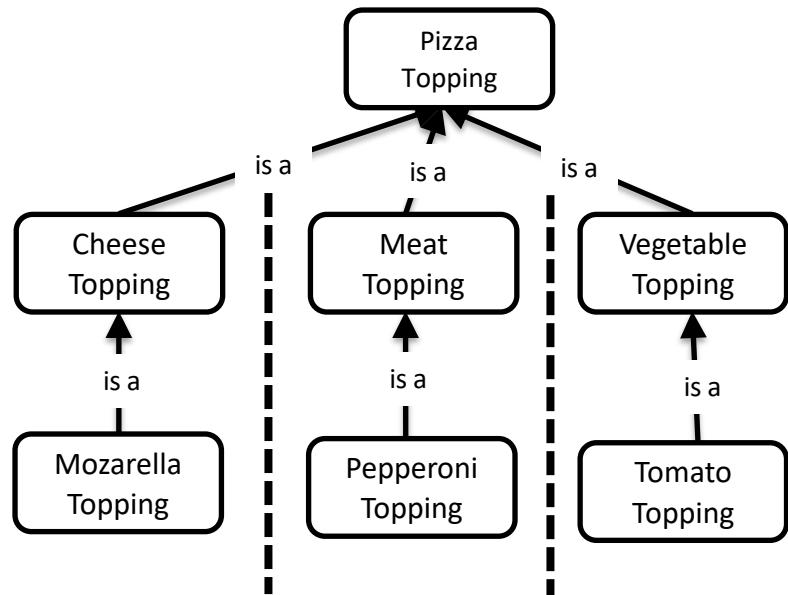
Hypothèse du monde ouvert

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.

Description: MargheritaPizza

Equivalent To +

SubClass Of +

● hasTopping **only** (MozarellaTopping or TomatoTopping)

● hasTopping **some** MozarellaTopping

● hasTopping **some** TomatoTopping

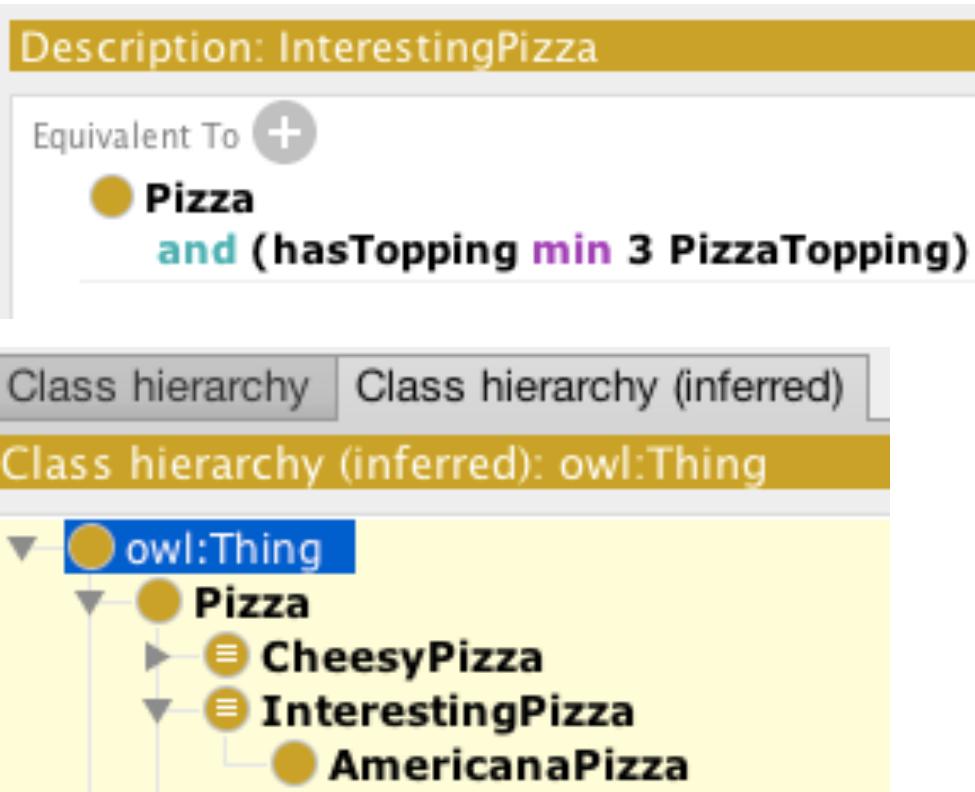
● NamedPizza

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: FourCheesePizza

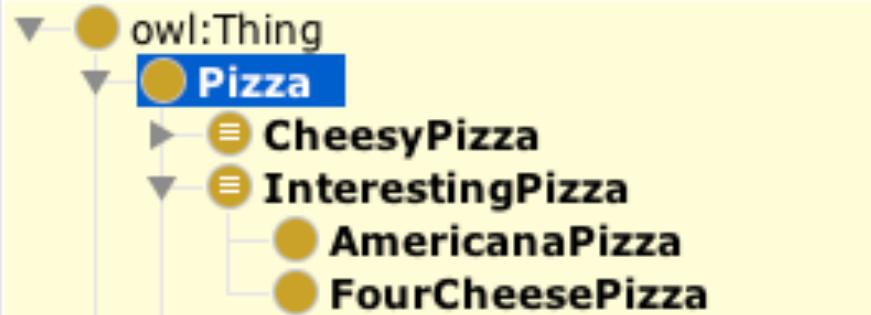
Equivalent To +

SubClass Of +

- hasTopping exactly 4 CheeseTopping
- NamedPizza

Class hierarchy Class hierarchy (inferred)

Class hierarchy (inferred): Pizza



Datatype properties et individus

The screenshot shows a semantic web editor interface. At the top, there are tabs: "Individuals by type", "Annotation property hierarchy", "Datatypes", "Object property hierarchy", and "Data property hierarchy". The "Data property hierarchy" tab is selected, and its sub-tab "owl:topDataProperty" is also selected. A green bar at the top of the main area says "Data property hierarchy: owl:topDataProperty". Below this, there are icons for creating new nodes and annotations. The main pane displays a tree structure with "owl:topDataProperty" expanded, showing its child "hasCalorificContentValue". On the right, there is a toolbar with a dropdown set to "Asserted" and a "Character" panel with a checked "Functional" checkbox.

The screenshot shows a semantic web editor interface. At the top, there are tabs: "Individuals by type", "Annotation", "Object property hierarchy", and "Datatypes". The "Individuals by type" tab is selected, and its sub-tab "Example-Margherita" is selected. A purple diamond icon indicates the current individual is "Example-Margherita". The main area is divided into two sections: "Description: Example-Margherita" and "Property assertions: Example-Margherita".
The "Description" section includes:

- "Types": A yellow circle icon labeled "MargheritaPizza" with a plus sign.
- "Same Individual As": A plus sign icon.
- "Different Individuals": A plus sign icon.

The "Property assertions" section includes:

- "Object property assertions": A plus sign icon.
- "Data property assertions": A blue plus sign icon. Below it, a green bar highlights the "hasCalorificCont" assertion, which has a value of "263".
- "Negative object property assertions": A plus sign icon.
- "Negative data property assertions": A plus sign icon.

Example-Margherita

Data Property



Asserted

Value

263

owl:topDataProperty

hasCalorificContentValue

Type xsd:integer

Lang

Cancel

OK

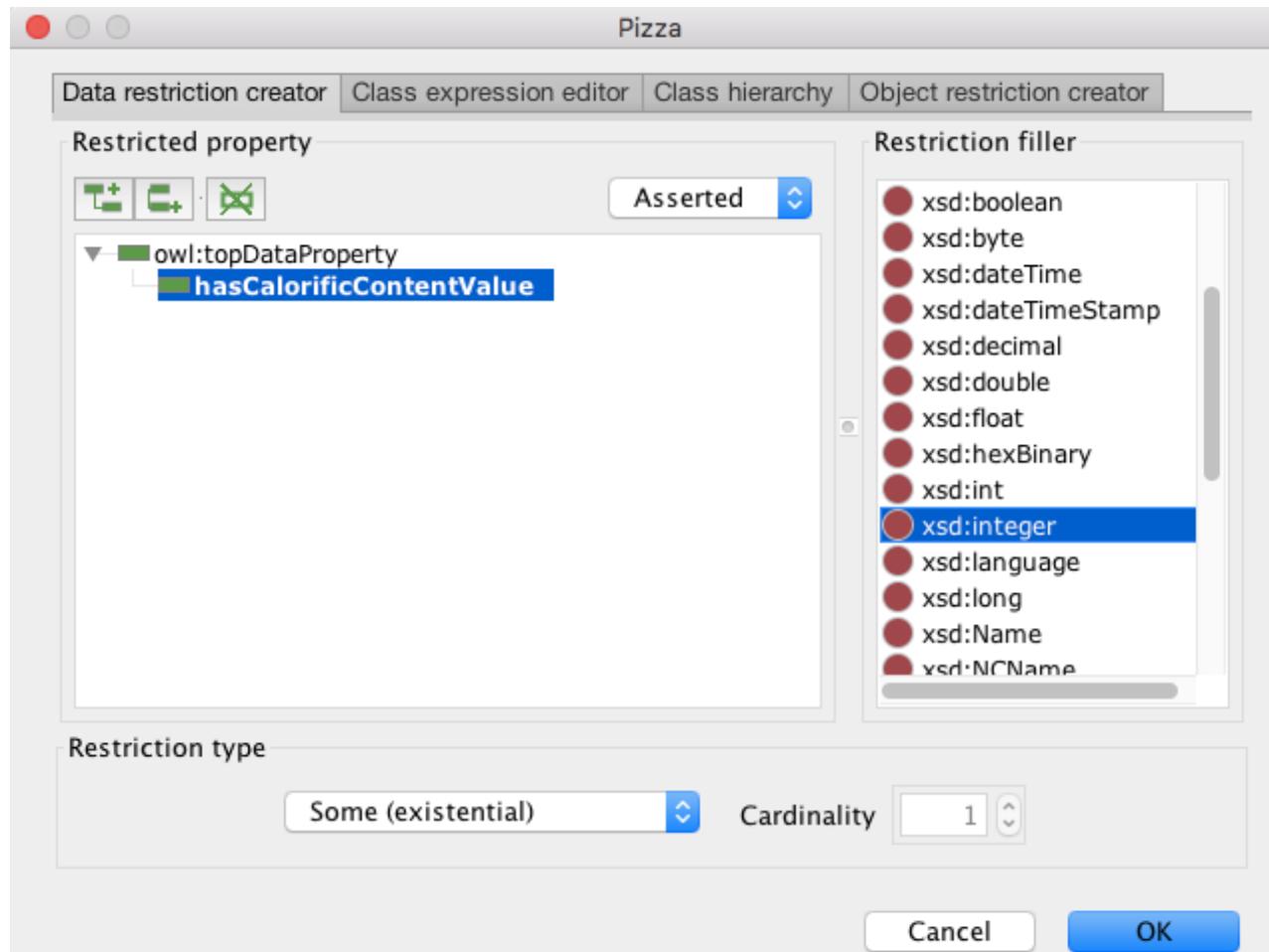
Axiome : Toutes les pizzas ont une valeur calorifique

Description: Pizza

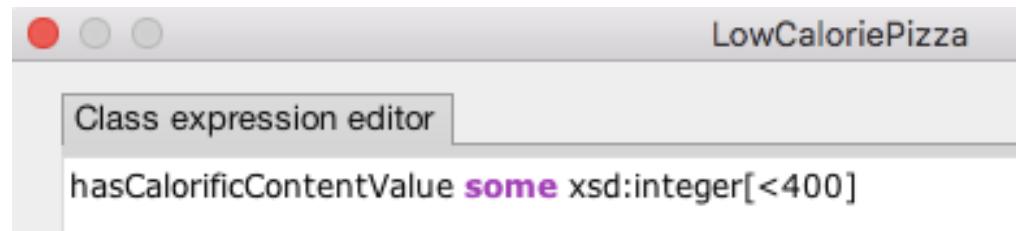
Equivalent To +

SubClass Of +

- hasBase some PizzaBase
- hasCalorificContentValue some xsd:integer



Raisonneur 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 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

Individuals by type: Country

The interface shows a tree structure under the heading "Individuals by type: Country". At the top level, there is a node for "Country (1)". Below it, "Italy" is shown as a child node. At the bottom level, "MargheritaPizza (1)" is shown, with "Example-Margherita" as a child node. There are three icons at the top: a purple diamond with a plus sign, a purple square with a crossed-out symbol, and a yellow circle with a plus sign.

- Country (1)
 - Italy
- MargheritaPizza (1)
 - Example-Margherita

Object property hierarchy Data property hierarchy

Object property hierarchy: hasCountryOfOrigin

The interface shows a tree structure under the heading "Object property hierarchy: hasCountryOfOrigin". At the top level, there is a node for "owl:topObjectProperty". Below it, "hasCountryOfOrigin" is shown as a child node. There are three icons at the top: a blue square with a plus sign, a blue square with a minus sign, and a blue square with a crossed-out symbol.

- owl:topObjectProperty
 - hasCountryOfOrigin

Description: MozarellaTopping

Equivalent To +

SubClass Of +

The interface shows a tree structure under the heading "Description: MozarellaTopping". At the top level, there is a node for "MozarellaTopping". Below it, "CheeseTopping" is shown as a child node. At the bottom level, "hasCountryOfOrigin value Italy" is shown. There are two buttons at the top: "Equivalent To" with a plus sign icon and "SubClass Of" with a plus sign icon.

- MozarellaTopping
 - CheeseTopping
 - hasCountryOfOrigin value Italy