

Introduction à OWL et Protégé

École d'été Interdisciplinaire en Numérique de la Santé (EINS 2025)

Adrien Barton^{1,2}, Paul Fabry²

¹ CNRS, IRIT, Université de Toulouse

² GRIIS, Université de Sherbrooke

28 mai 2025



GRIIS

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, showing a 'Class hierarchy' view for the 'drug administration specification' class.

The 'Class hierarchy' view on the left shows a tree structure of classes. The 'drug administration specification' class is highlighted, showing its subclasses: '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'.

The 'Annotations' view on the right shows the 'drug administration specification' class with the following annotations:

- 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 :

The 'Description' view on the right shows the 'drug administration specification' class with the following descriptions:

- 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**: +

The bottom of the interface shows the 'Object property hierarchy' view, which is currently empty. The bottom status bar indicates 'To use the reasoner click Reasoner->Start reasoner' and 'Show Inferences' is checked.

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

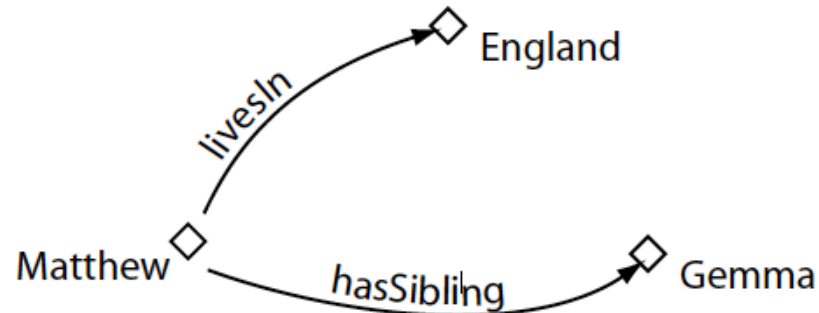


Figure 3.2: Representation Of Properties

Classes

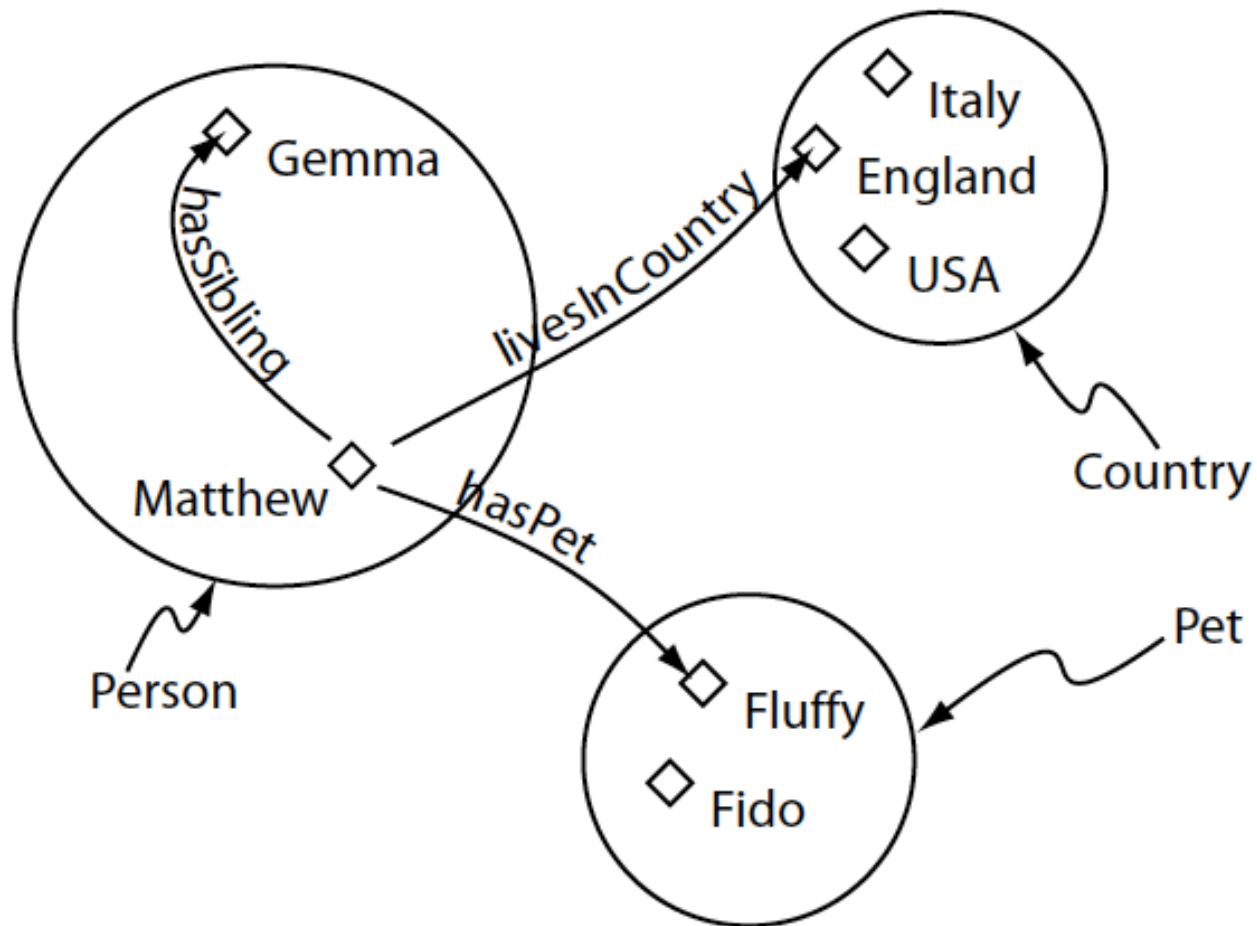
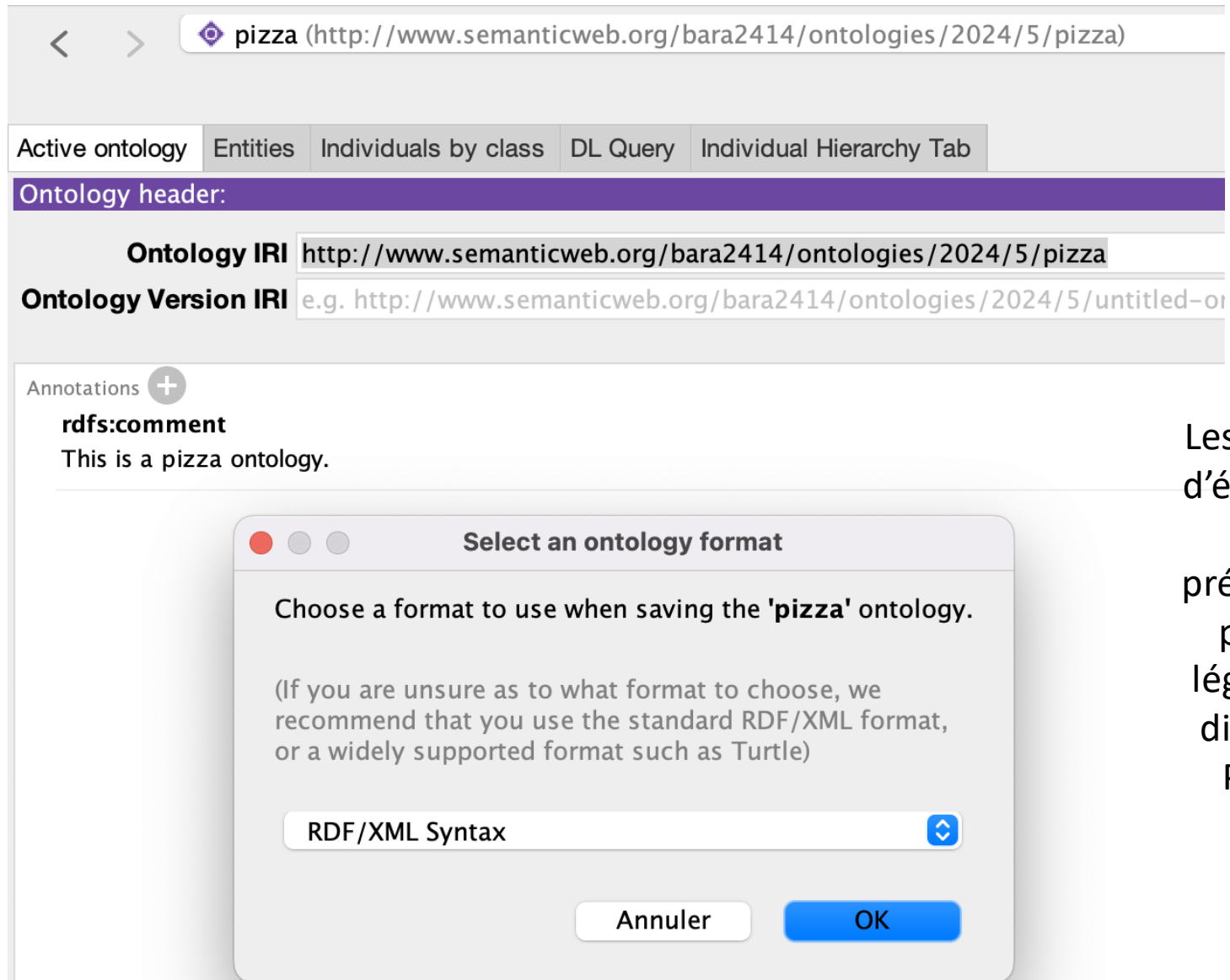


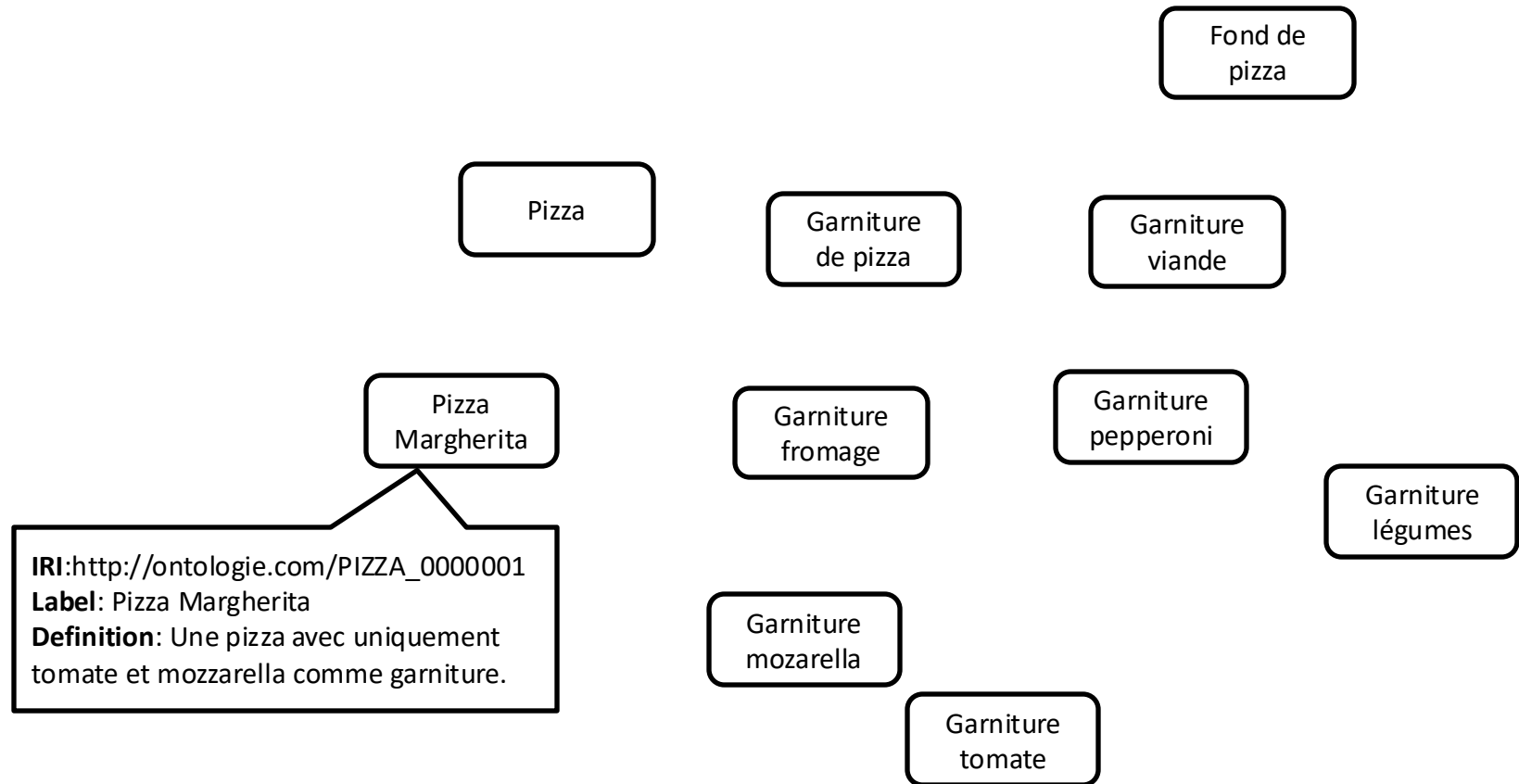
Figure 3.3: Representation Of Classes (Containing Individuals)

Créer une ontologie sous Protégé 5.x

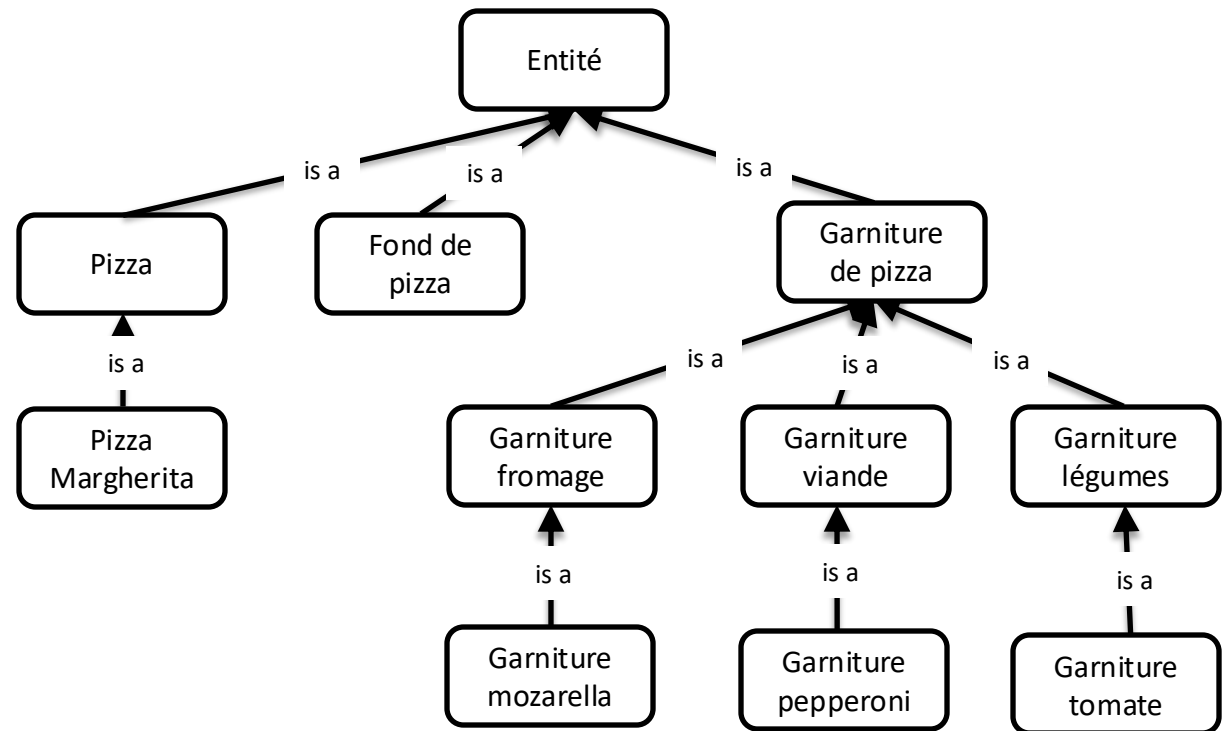


Les captures d'écran dans cette présentation peuvent légèrement différer de Protégé 5.5.0.

Quelles classes pour une ontologie des pizzas ?



Une première taxonomie des pizzas



Une première taxonomie des pizzas

Entité

a comme
fond

```
<!-- 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>
```

ture
mes

s a

ture
ate

Premières classes

Active ontology × Entities × Individuals by class

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Class hierarchy: Pizza ? || = □ ×

Asserted

- owl:Thing
 - Pizza**
 - PizzaBase
 - PizzaTopping

Description: Pizza

- Equivalent To +
- SubClass Of +
- General class axioms +
- SubClass Of (Anonymous Ancestor)
- Instances +
- Target for Key +
- Disjoint With +
- Disjoint Union Of +

Pizza

Class hierarchy Expression editor

Asserted

- owl:Thing
 - Pizza**
 - PizzaBase**
 - PizzaTopping**

Annuler OK

Créer des taxonomies de classes

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

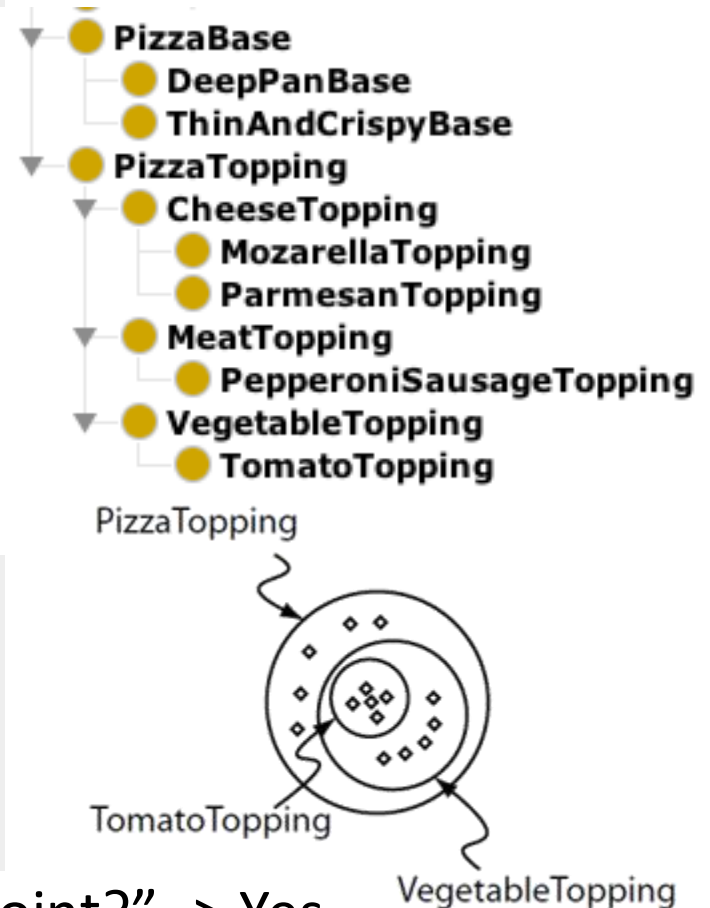
Enter hierarchy

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

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

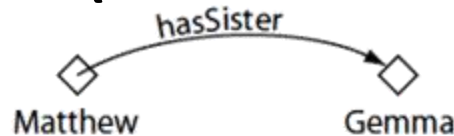
Prefix

Suffix

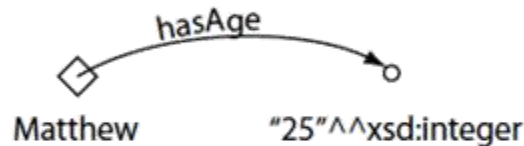


“Do you want to make sibling classes disjoint?” -> Yes

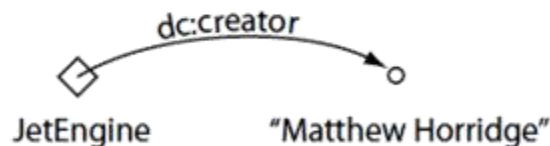
Trois types de propriétés (aka relations)



An object property linking the individual Matthew to the individual Gemma



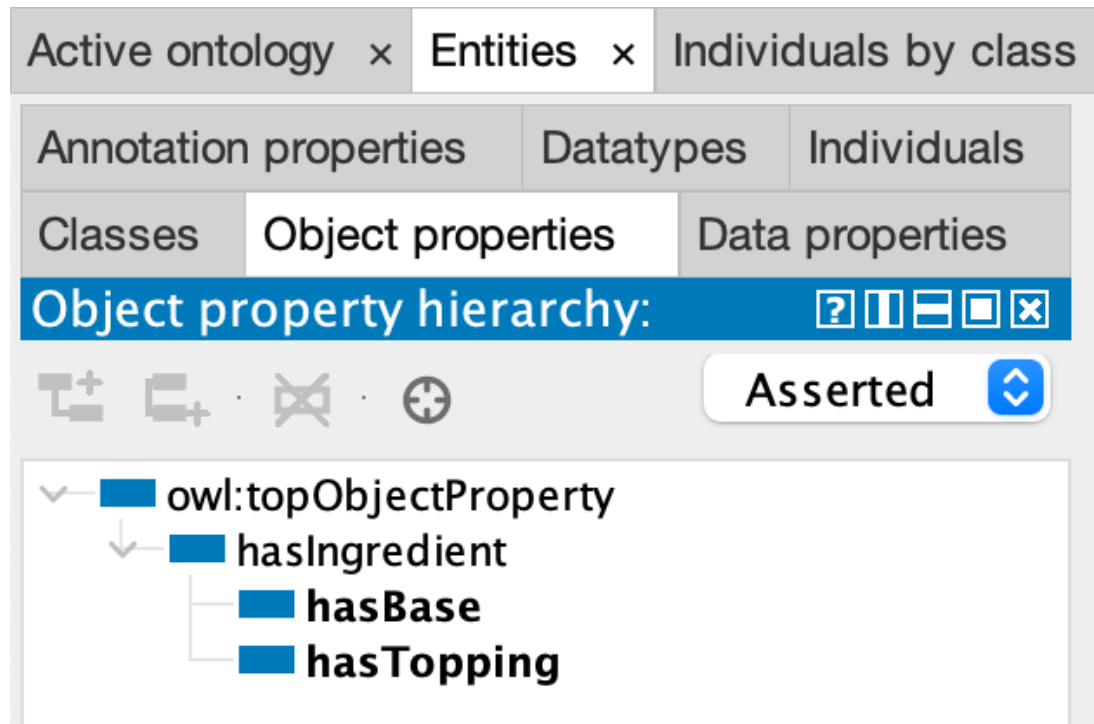
A datatype property linking the individual Matthew to the data literal '25', which has a type of an xsd:integer.



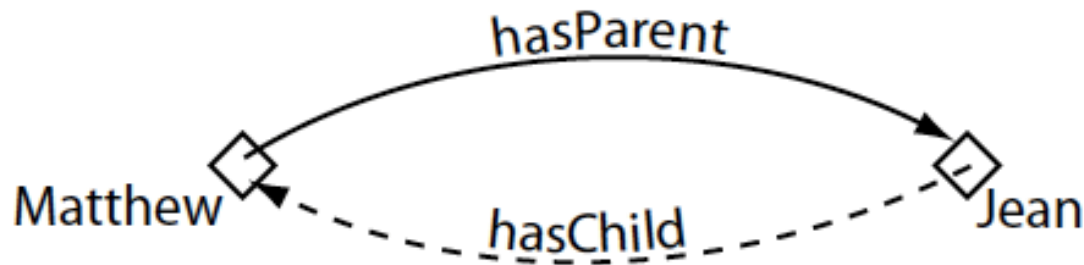
An annotation property, linking the class 'JetEngine' to the data literal (string) "Matthew Horridge".

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

hasIngredient

Asserted

owl:topObjectProperty

- hasIngredient**
- isIngredientOf**

Cancel OK

Description: **hasIngredient**

Equivalent To +

SubProperty Of +

Inverse Of +

Domains (intersection) +

Ranges (intersection) +

Disjoint With +

SuperProperty Of (Chain) +

Caractéristique des relations

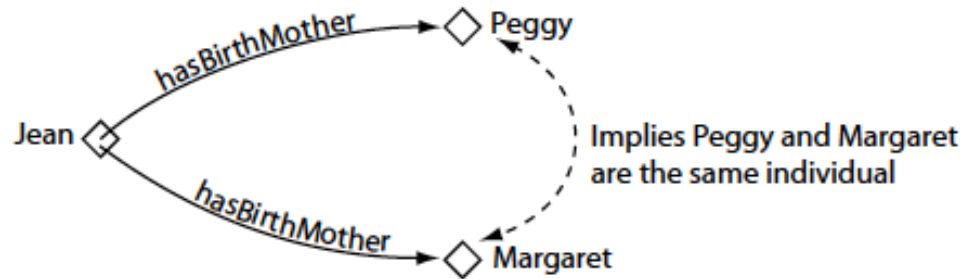


Figure 4.18: An Example Of A Functional Property: `hasBirthMother`

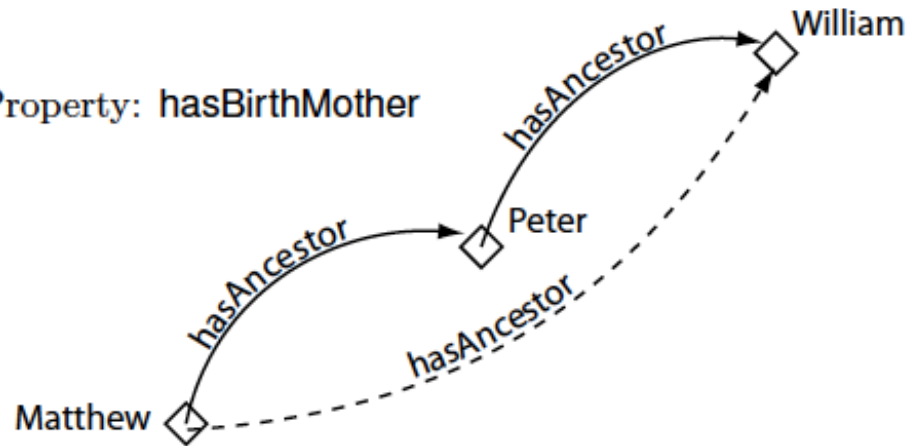


Figure 4.20: An Example Of A Transitive Property: `hasAncestor`

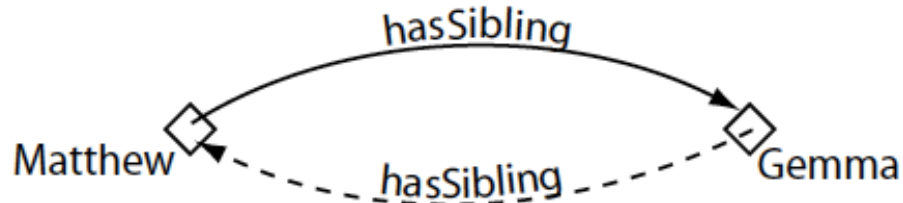


Figure 4.21: An Example Of A Symmetric Property: `hasSibling`

Caractéristiques des relations : application

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

The screenshot shows a web application interface for configuring an ontology. The top navigation bar includes tabs: 'Active ontology', 'Entities', 'Individuals by class', 'Individual Hierarchy Tab', and 'DL Query'. Below this, there are tabs for 'Annotation properties', 'Datatypes', and 'Individuals'. The 'Object properties' tab is selected, showing a hierarchy of properties under 'owl:topObjectProperty'. The 'hasIngredient' property is highlighted. To the right, the 'Characteristics' panel for 'hasIngredient' is displayed, showing a list of checkboxes for property characteristics. The 'Transitive' checkbox is checked, while others are unchecked.

Active ontology x Entities x Individuals by class x Individual Hierarchy Tab x DL Query x

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Object property hierarchy: hasIngredient ? ? ? ? ?

Characteristics: hasIngredient ? ? ? ? ?

Asserted

- ☐ Functional
- ☐ Inverse functional
- ☒ Transitive
- ☐ Symmetric
- ☐ Asymmetric
- ☐ Reflexive
- ☐ Irreflexive

owl:topObjectProperty

- isIngredientOf
 - isBaseOf
 - isToppingOf
- hasIngredient
 - hasBase
 - hasTopping

Domain & Range

The screenshot shows a web ontology editor interface. At the top, there are tabs for 'Active ontology', 'Entities', 'Individuals by class', 'Individual Hierarchy Tab', and 'DL Query'. Below these are tabs for 'Classes', 'Object properties', 'Data properties', 'Annotation properties', 'Datatypes', and 'Individuals'. The 'Object properties' tab is selected, and the 'Object property hierarchy: hasTopping' is displayed. The hierarchy shows 'owl:topObjectProperty' as the root, with children 'isIngredientOf' and 'hasIngredient'. 'isIngredientOf' has children 'isBaseOf' and 'isToppingOf'. 'hasIngredient' has children 'hasBase' and 'hasTopping'. The 'hasTopping' property is highlighted. To the right of the hierarchy, there is a button labeled 'Asserted'. On the far right, there are two sections: 'Domains (intersection)' with a yellow circle and the text 'Pizza', and 'Ranges (intersection)' with a yellow circle and the text 'PizzaTopping'.

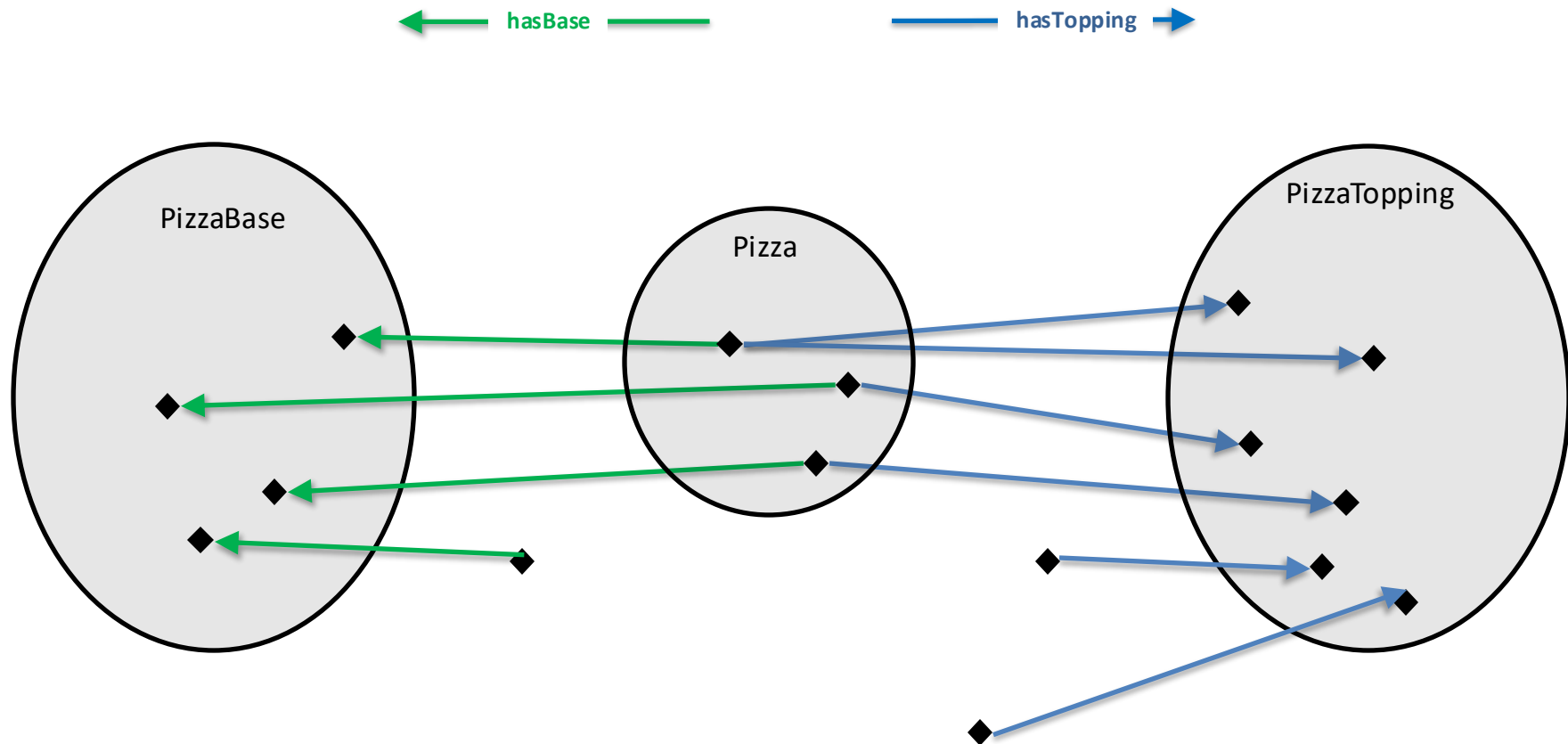
hasBase: Domain Pizza, Range PizzaBase (utiliser autocomplete)



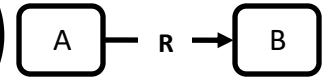
Property Domains And Ranges In OWL — It is important to realise that in OWL domains and ranges should *not* be viewed as constraints to be checked. They are used as ‘axioms’ in reasoning. For example if the property **hasTopping** has the domain set as **Pizza** and we then applied the **hasTopping** 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**

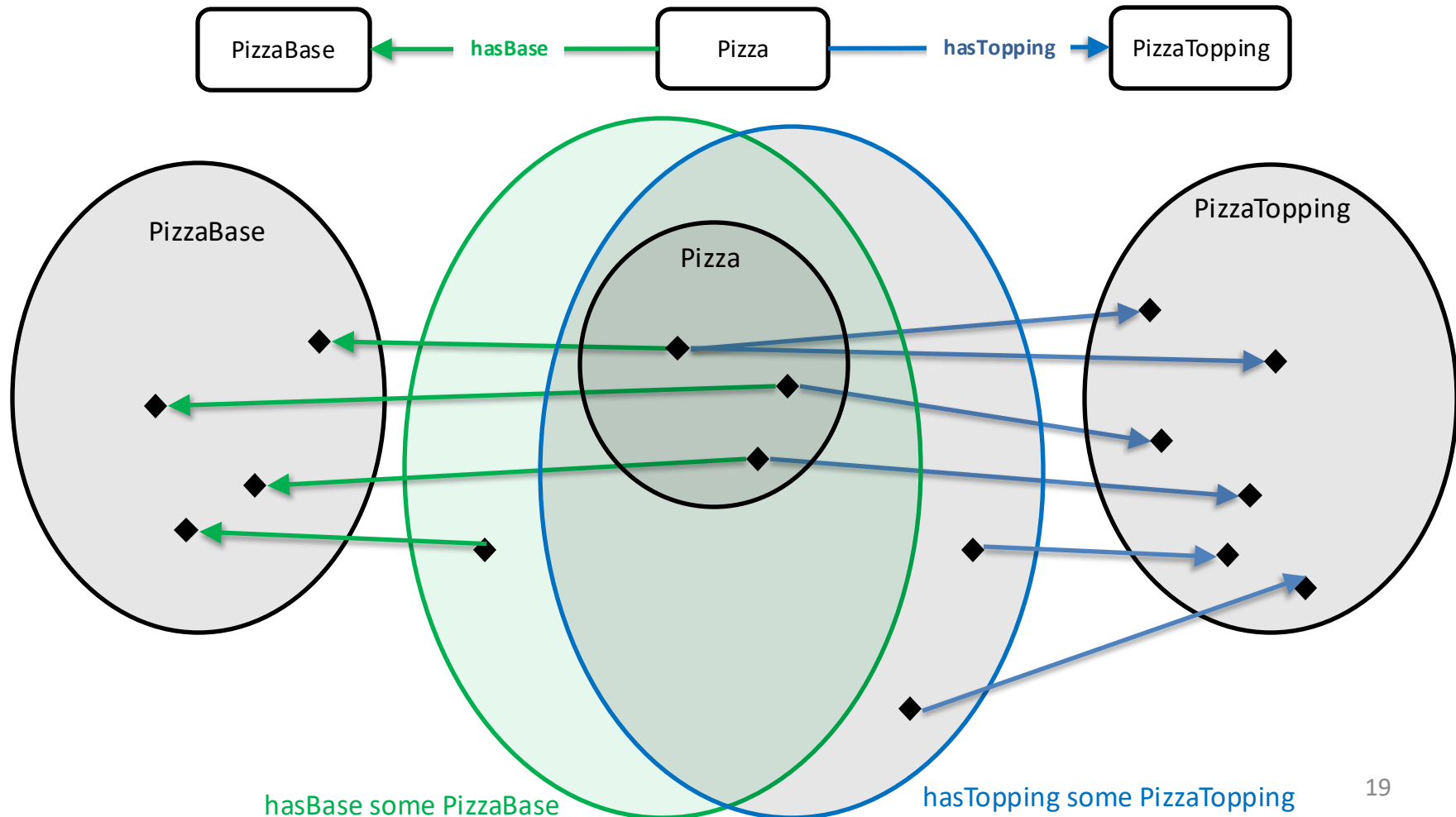
Axiomes existentiels (restrictions existentielles)



Axiomes existentiels (restrictions existentielles)

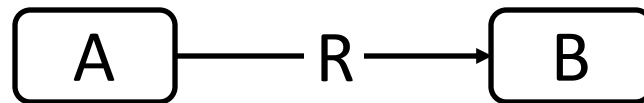


A SubclassOf R some B



Axiomes existentiels (restrictions existentielles)

Un axiome est une restriction de classe



A SubClassOf (R *some* B)



classe « anonyme »

SubClass Of +

● R some B

Axiomes existentiels (restrictions existentielles)

1

Description: **Pizza**

Equivalent To +

SubClass Of +

- hasBase some PizzaBase

2

Description: **NamedPizza**

Equivalent To +

SubClass Of +

- Pizza

3

Description: **MargheritaPizza**

Equivalent To +

SubClass Of +

- hasTopping some MozzarellaTopping
- hasTopping some TomatoTopping
- NamedPizza

General class axioms +

SubClass Of (Anonymous Ancestor)

- hasBase some PizzaBase

4

Description: **AmericanaPizza**

Equivalent To +

SubClass Of +

- hasTopping some MozzarellaTopping
- hasTopping some PepperoniSausageTopping
- 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

Raisonneur : détecter les incohérences

Description: **ProbelInconsistentTopping**

Equivalent To +

SubClass Of +

● **CheeseTopping**

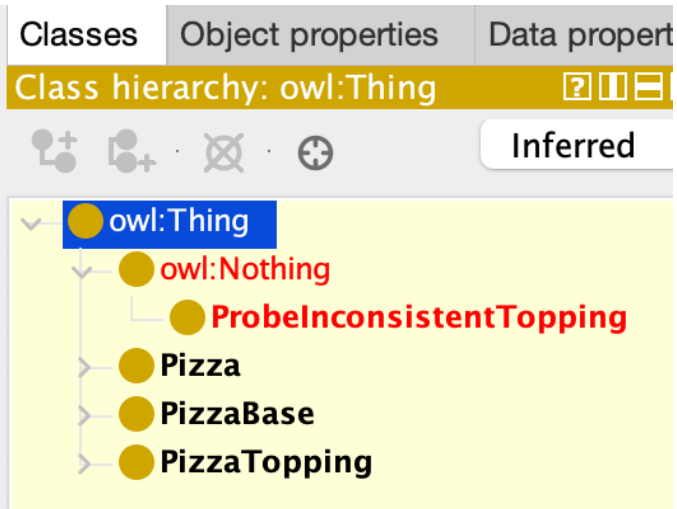
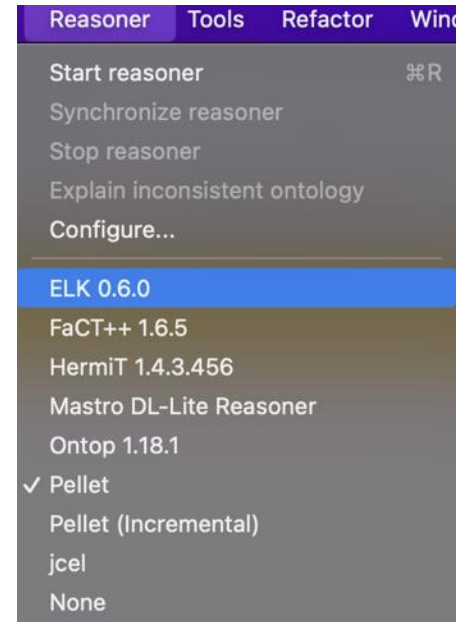
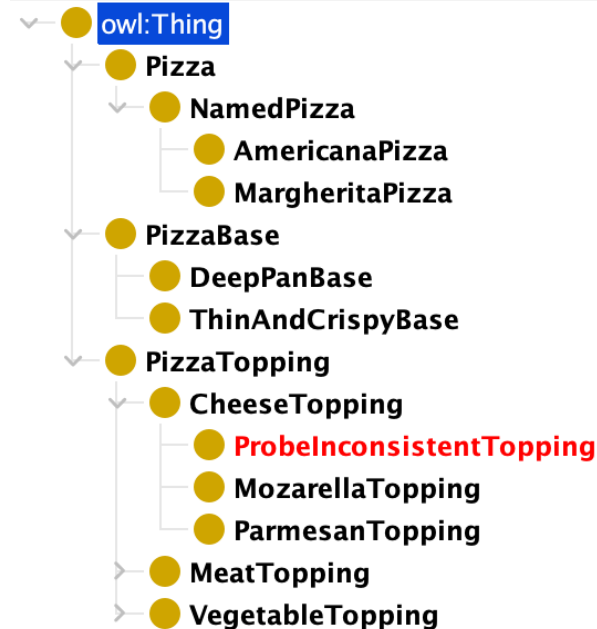
● **VegetableTopping**

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Class hierarchy: owl:Thing ? II = □ ×

Asserted



Ne pas oublier d'arrêter le raisonneur.
Réessayer (start / synchronise) 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

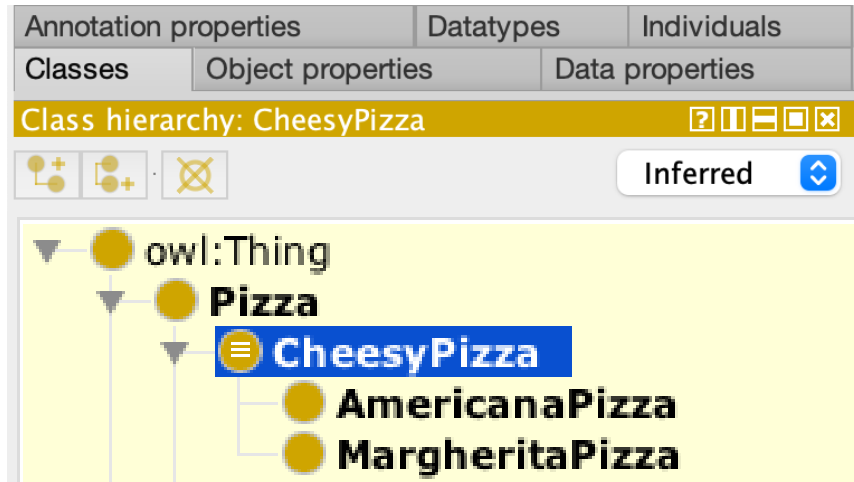
Description: CheesyPizza

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))




Hypothèse du monde ouvert

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

Description: MargheritaPizza

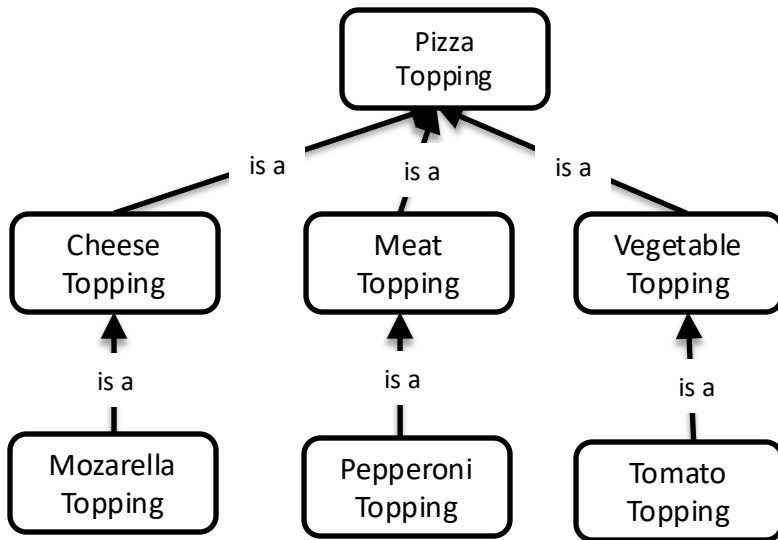
Equivalent To 

SubClass Of 

-  hasTopping **some** MozzarellaTopping
-  hasTopping **some** TomatoTopping
-  NamedPizza

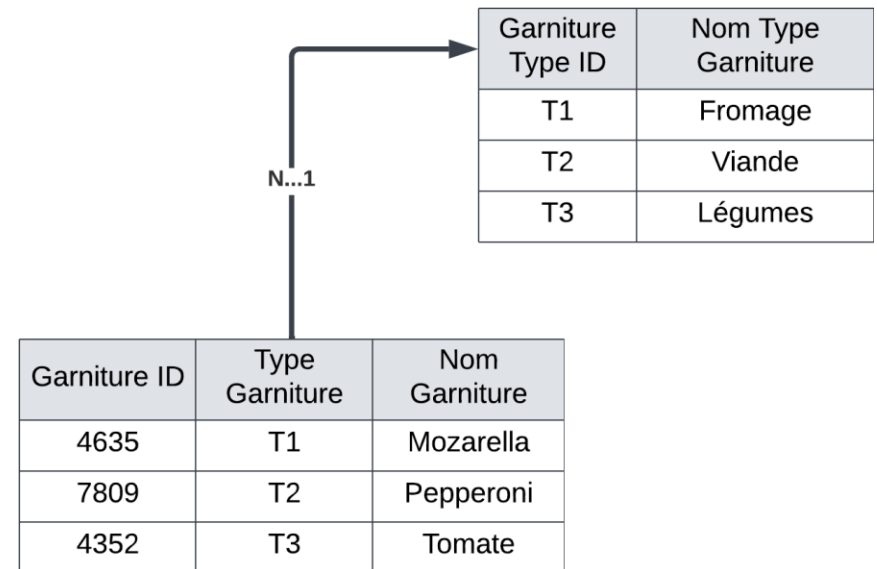
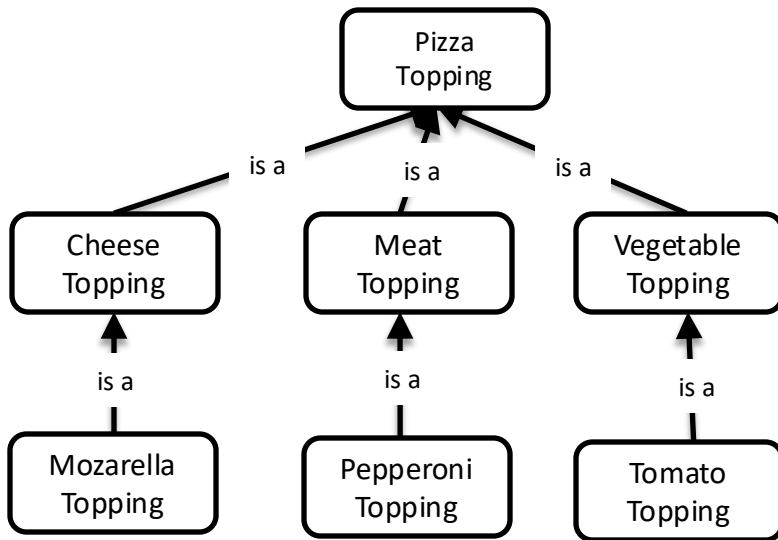
- « *Ce qui n'est pas énoncé peut être vrai ou faux* »
- Nécessité de contraindre l'ontologie :
 - Une MozzarellaTopping pourrait être un MeatTopping
→ nécessité de disjonctions
 - Une MargheritaPizza pourrait avoir un MeatTopping
→ nécessité de restrictions universelles

Hypothèse du monde ouvert



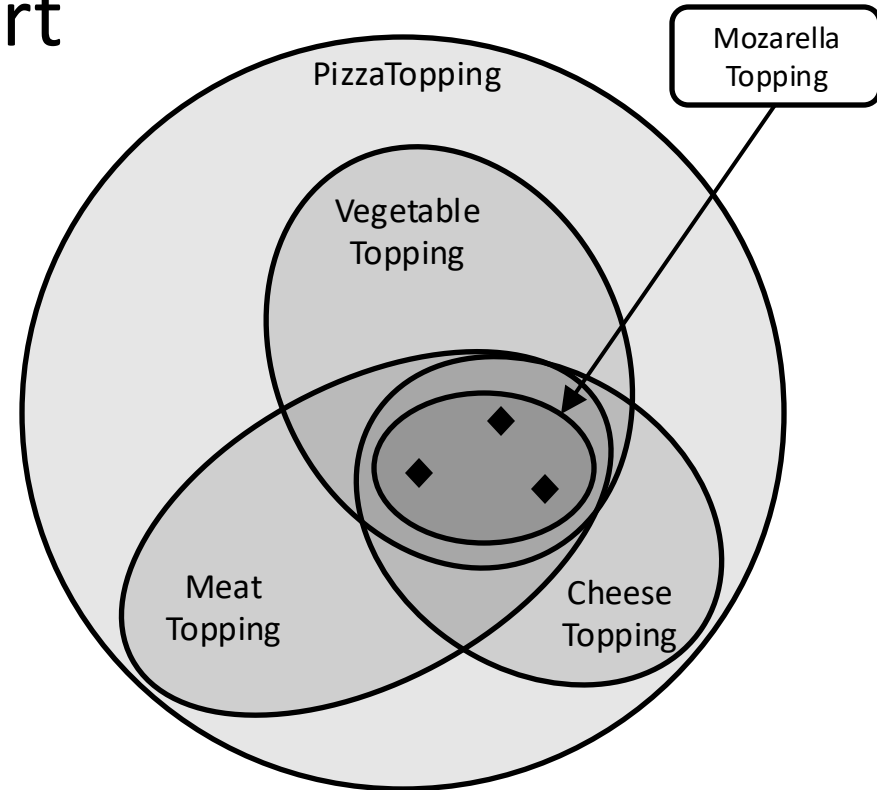
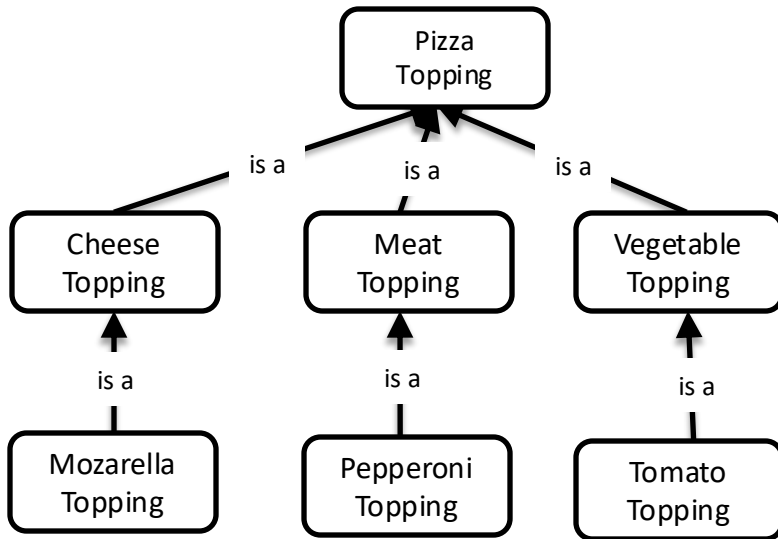
Hypothèse du monde ouvert

BD : monde fermé



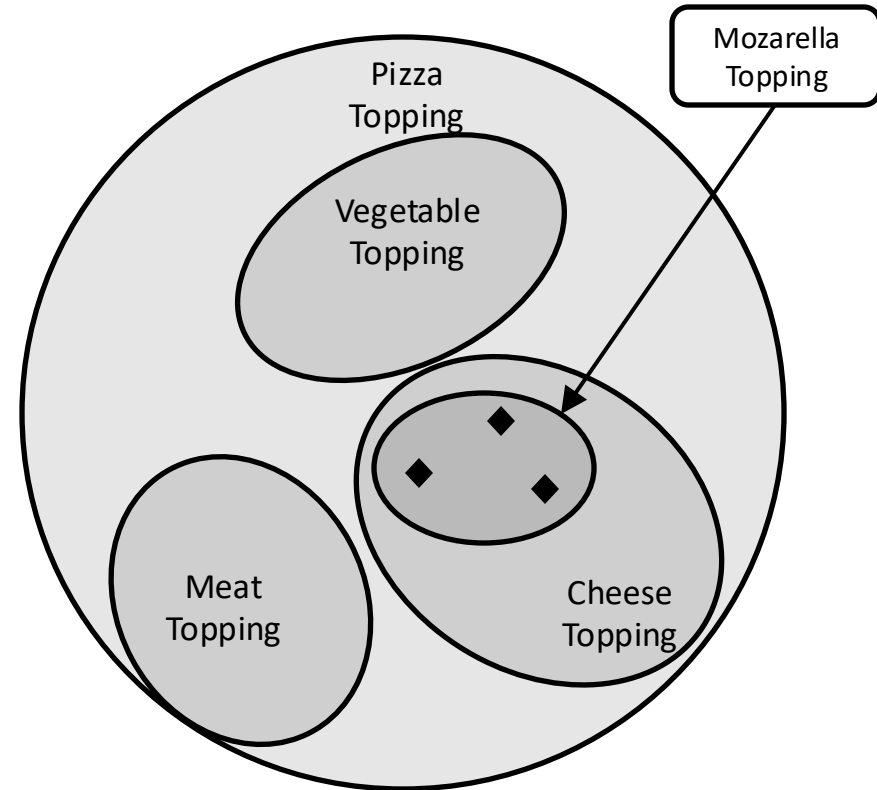
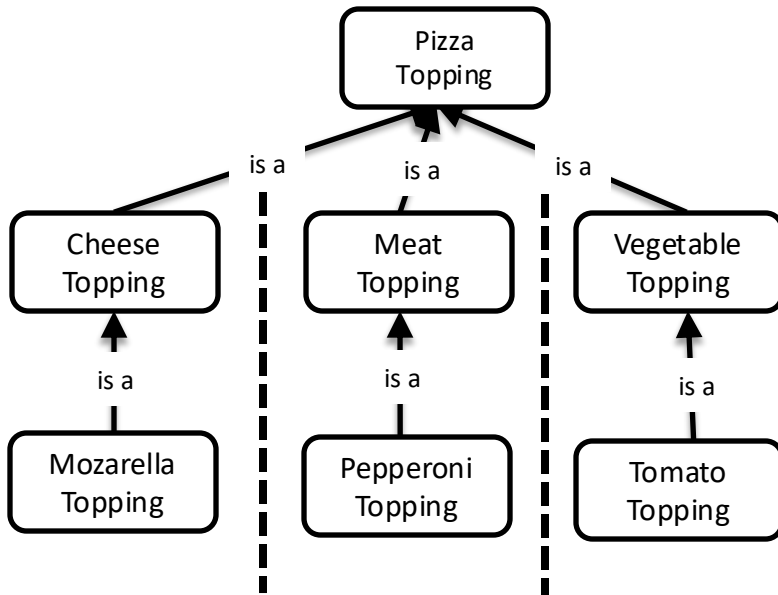
Hypothèse du monde ouvert

Ontologie : monde ouvert



Hypothèse du monde ouvert

Disjonction



Restriction universelle

- 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

1.a

Créer InterestingPizza

Description: InterestingPizza

Equivalent To

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

1.b

Start (or synchronize) reasoner

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Class hierarchy: owl:Thing

Inferred

- owl:Thing
 - Pizza**
 - CheesyPizza
 - InterestingPizza
 - AmericanaPizza

2.a

Créer FourCheesePizza

Description: FourCheesePizza

Equivalent To

SubClass Of

- hasTopping **exactly 4** CheeseTopping
- NamedPizza

2.b

Start (or synchronize) reasoner

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Class hierarchy: owl:Thing

Inferred

- owl:Thing
 - Pizza**
 - CheesyPizza
 - InterestingPizza
 - AmericanaPizza
 - FourCheesePizza

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

Active ontology Entities Individuals by class

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Individuals: Example-Margherita

Example-Margherita

Example-Margherita

Data Property

Asserted

owl:topDataProperty

hasCalorificContentValue

Value

263

Language Tag

Language Tag

Datatype

xsd:decimal

Annuler OK

Description: Example-Margherita

Property assertions: Example-Margherita

Types

MargheritaPizza

Same Individual As

Object property assertions

Data property assertions

hasCalorificContentValue 263

Axiome : Toutes les pizzas ont une valeur calorifique

Description: Pizza

Equivalent To +

SubClass Of +

● hasBase **some** PizzaBase

● hasCalorificContentValue **some** xsd:integer

Pizza

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

Restricted property

owl:topDataProperty
hasCalorificContentValue

Assertion

Restriction filler

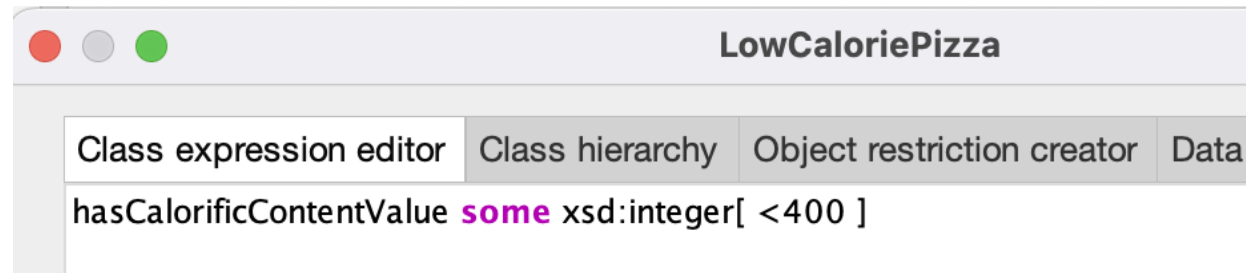
- xsd:hexBinary
- xsd:int
- xsd:integer
- xsd:language
- xsd:long
- xsd:Name
- xsd:NCName
- xsd:negativeInteger
- xsd:NMTOKEN
- xsd:nonNegativeInteger
- xsd:nonPositiveInteger
- xsd:normalizedString
- xsd:positiveInteger
- xsd:short

Restriction type


Some (existential) Cardinality 1


Annuler 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 qu'une garniture mozzarella a comme pays d'origine l'Italie.
- Les pays sont-ils des classes ou des individus?

Axiome reliant toutes les instances d'une classe au même individu

Active ontology | Entities | Individuals by class

Direct instances: Italy [?] [||] [≡] [□] [✕]

For: ● Country

◆ Italy

Annotation properties | Datatypes | Individuals

Classes | Object properties | Data properties

Object property hierarchy: hasCoun [?] [||] [≡] [□] [✕]

Asserted [↕]

- owl:topObjectProperty
 - hasCountryofOrigin
 - hasIngredient
 - isIngredientOf

Description: MozzarellaTopping

Equivalent To [⊕]

SubClass Of [⊕]

- CheeseTopping
- hasCountryofOrigin value Italy

Questions ?

École d'été Interdisciplinaire en Numérique de la Santé (EINS 2025)

Adrien Barton^{1,2}, Paul Fabry²

¹ CNRS, IRIT, Université de Toulouse

² GRIIS, Université de Sherbrooke

28 mai 2025



GRIIS