

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, Université de Toulouse

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 includes:
 - 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

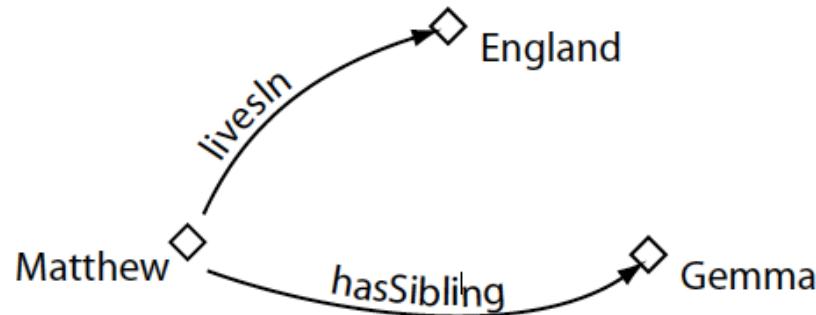


Figure 3.2: Representation Of Properties

Classes

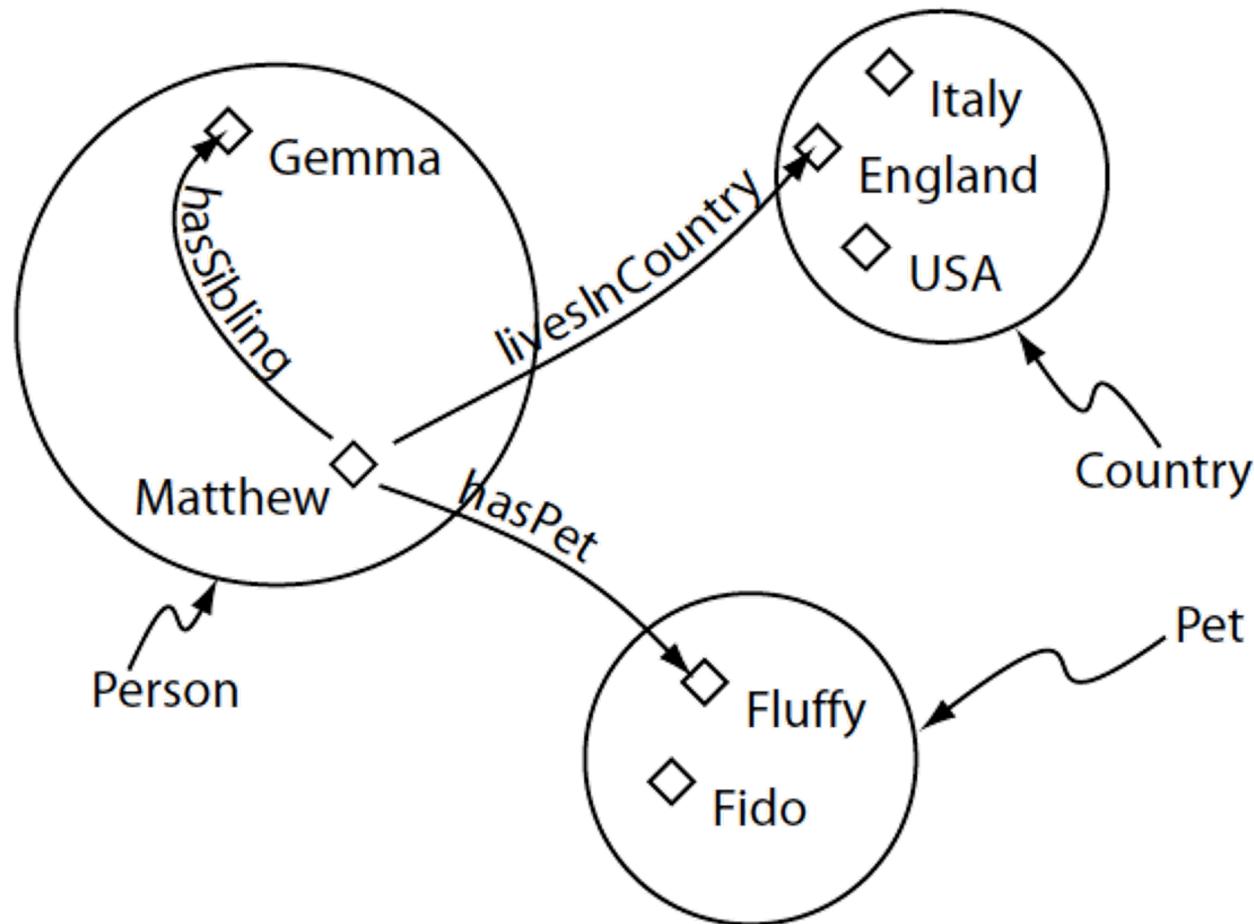


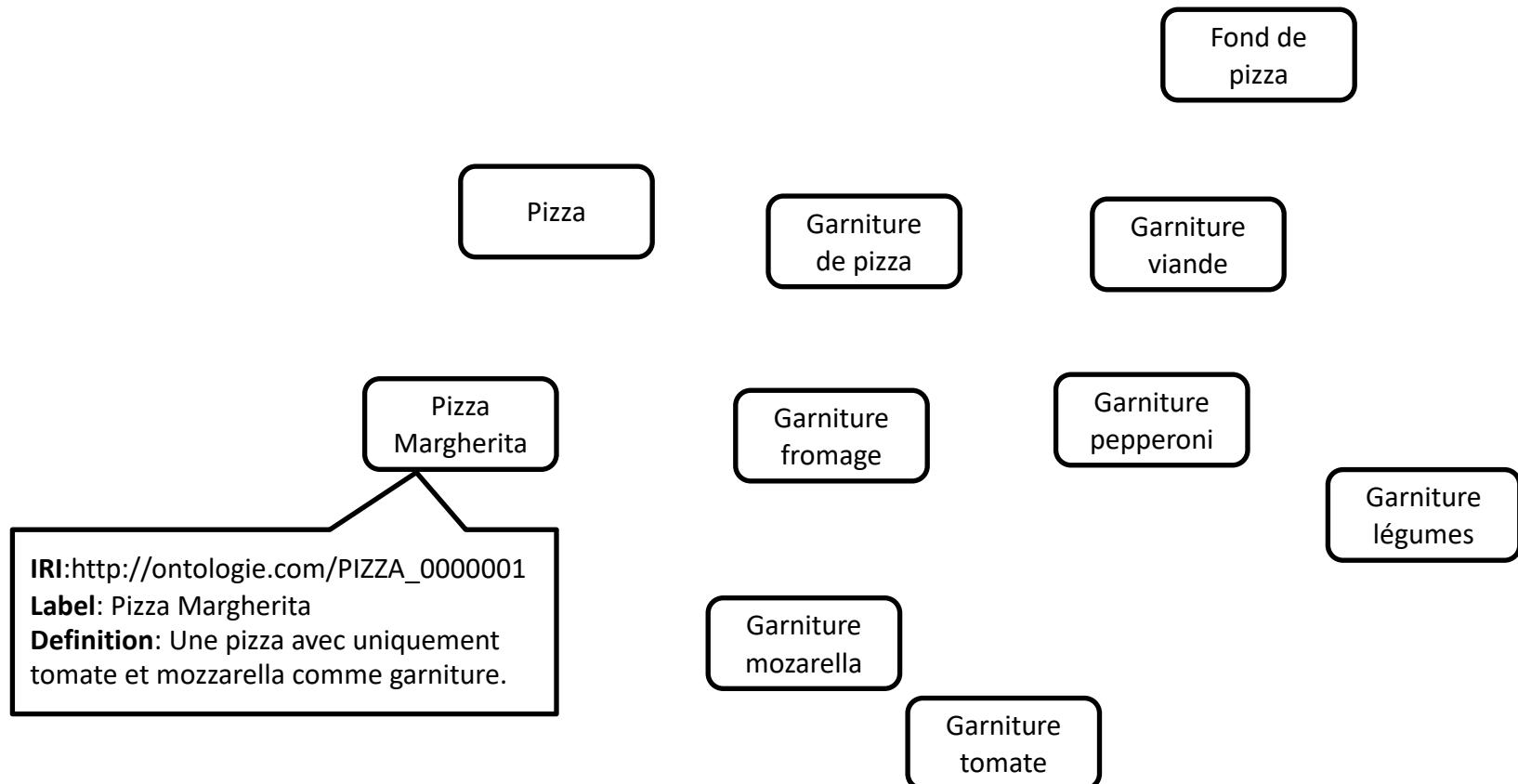
Figure 3.3: Representation Of Classes (Containing Individuals)

Créer une ontologie sous Protégé 5.x

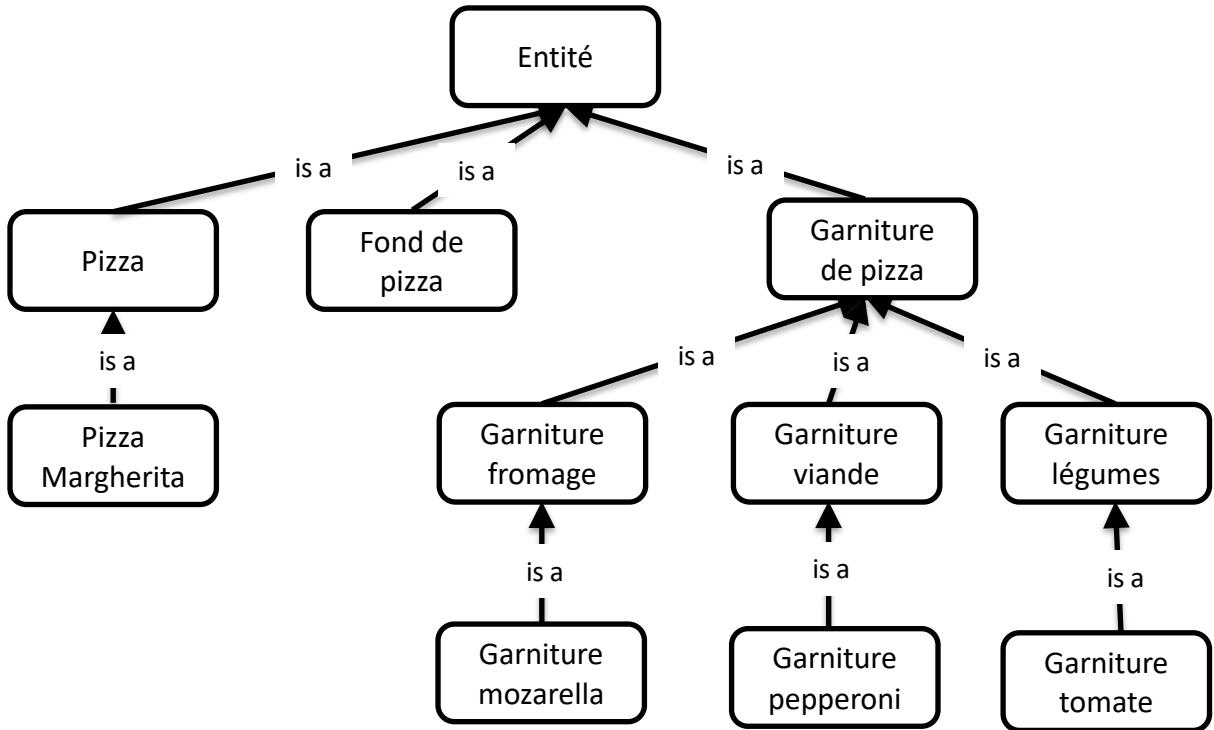
The screenshot shows the Protégé 5.x interface with the following details:

- Toolbar:** Includes back, forward, and search icons.
- Title Bar:** Shows the ontology name: **pizza** (<http://www.semanticweb.org/barabara2414/ontologies/2024/5/pizza>)
- Tab Bar:** Active ontology (highlighted), Entities, Individuals by class, DL Query, Individual Hierarchy Tab.
- Ontology header:** Contains fields for Ontology IRI (<http://www.semanticweb.org/barabara2414/ontologies/2024/5/pizza>) and Ontology Version IRI (e.g. <http://www.semanticweb.org/barabara2414/ontologies/2024/5/untitled-ontology-1>).
- Annotations:** A section containing an **rdfs:comment** annotation: "This is a pizza ontology."
- Save Format Dialog:** A modal window titled "Select an ontology format" with the instruction "Choose a format to use when saving the 'pizza' ontology." It includes a note: "(If you are unsure as to what format to choose, we recommend that you use the standard RDF/XML format, or a widely supported format such as Turtle)." The selected format is "RDF/XML Syntax".
- Buttons:** Annuler (Cancel) and OK.

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

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Class hierarchy: Pizza ? I ≡ □ X

+ + ✖ + 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 +

OK Annuler

+ + ✖ + Asserted ▼

owl:Thing
|
+--- Pizza
|
+--- PizzaBase
|
+--- PizzaTopping

Créer des taxonomies de classes

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

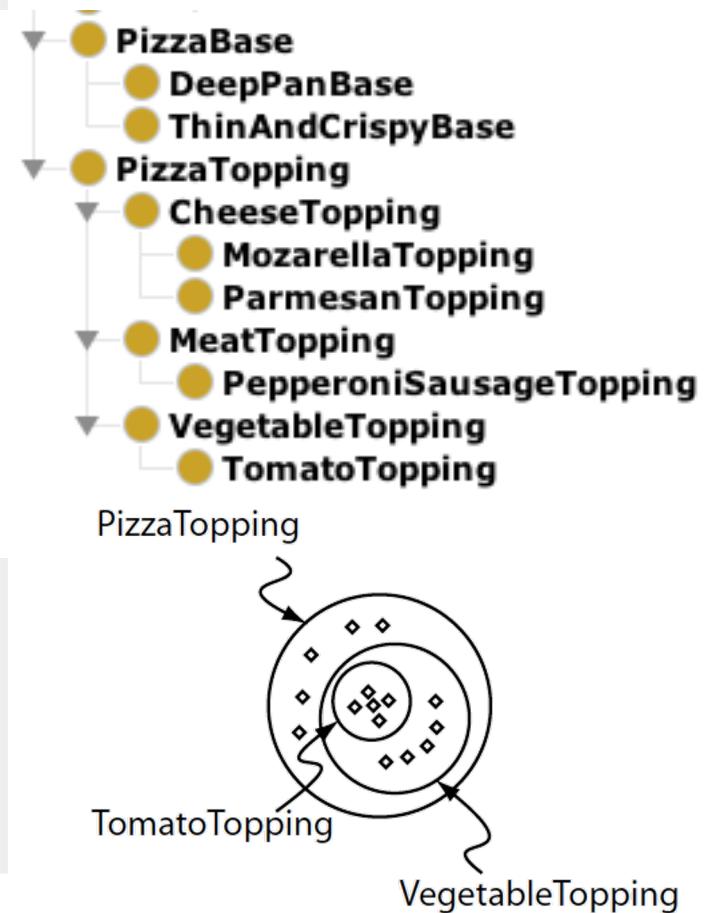
Enter hierarchy

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

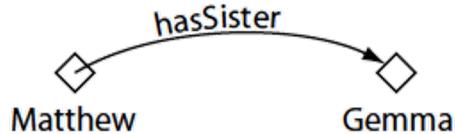
```
Cheese
    Mozarella
    Parmesan
Meat
    PepperoniSausage
Vegetable
    Tomato
```

Prefix

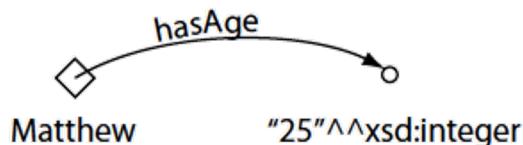
Suffix Topping



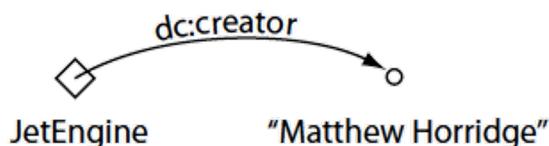
Trois types de propriétés (aka relations)



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.



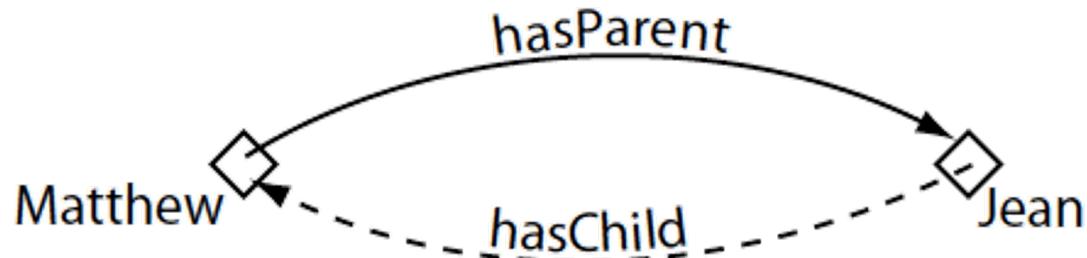
An annotation property, linking the class 'JetEngine' to the data literal (string) "Matthew Horridge".

Figure 4.12: The Different types of OWL Properties

Créer des ‘object properties’

The screenshot shows the Protégé ontology editor interface. The top navigation bar includes 'Active ontology' (with a close button), 'Entities' (with a close button), and 'Individuals by class'. Below this is a secondary navigation bar with tabs for 'Annotation properties', 'Datatypes', and 'Individuals', with 'Object properties' selected. Underneath are tabs for 'Classes', 'Object properties' (selected), and 'Data properties'. A blue header bar displays 'Object property hierarchy:' followed by icons for help, search, copy, paste, and delete. Below this is a toolbar with icons for adding a class, adding an individual, deleting, and merging. To the right of the toolbar is a dropdown menu set to 'Asserted' with a sorting icon. The main content area shows a tree view of object properties. At the top level is 'owl:topObjectProperty', which has three children: 'hasIngredient', 'hasBase', and 'hasTopping'. Each of these properties is represented by a blue square icon.

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

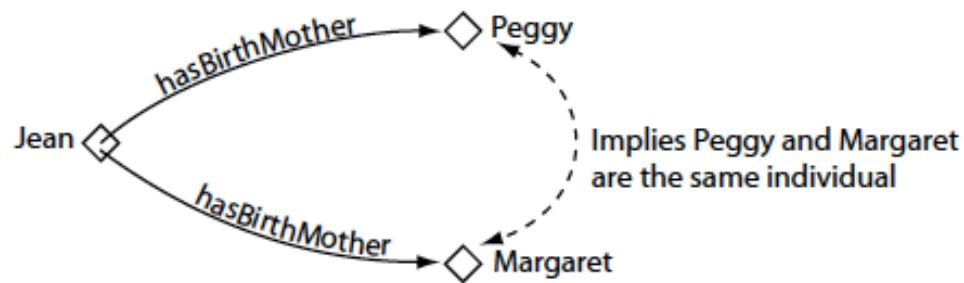


Figure 4.18: An Example Of A Functional Property: `hasBirthMother`

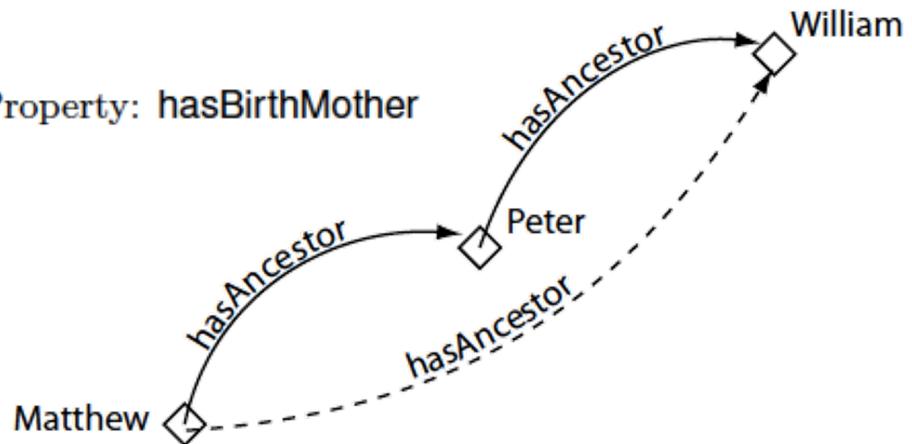


Figure 4.20: An Example Of A Transitive Property: `hasAncestor`

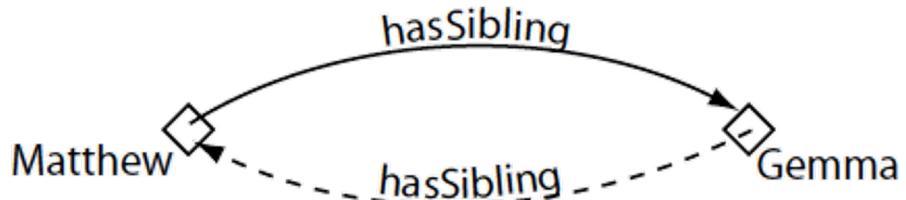


Figure 4.21: An Example Of A Symmetric Property: `hasSibling`

Caractéristiques des relations : application

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

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

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Object property hierarchy: has [?][I][F][S][X]

Annotations Usage

Characteristics: hasIngredient [?][I][F][S][X]

Asserted

Functional

Inverse functional

Transitive

Symmetric

Asymmetric

Reflexive

Irreflexive

The screenshot shows the Protege ontology editor interface. The top navigation bar includes tabs for Active ontology, Entities, Individuals by class, Individual Hierarchy Tab, and DL Query. Below the tabs are tabs for Annotation properties, Datatypes, and Individuals, followed by Classes, Object properties (which is selected), and Data properties. A blue header bar displays 'Object property hierarchy: has' with icons for question mark, inverse, functional, symmetric, and asserted status. To the right of this is a blue header bar for 'Characteristics: hasIngredient' with the same set of icons. The main workspace shows a tree view of object properties under 'owl:topObjectProperty'. The 'hasIngredient' property is highlighted in blue. Other properties listed include isIngredientOf, isBaseOf, isToppingOf, hasBase, and hasTopping. On the right side, a panel titled 'Characteristics: hasIngredient' lists nine checkboxes corresponding to the icons in the header: Functional (unchecked), Inverse functional (unchecked), Transitive (checked), Symmetric (unchecked), Asymmetric (unchecked), Reflexive (unchecked), and Irreflexive (unchecked). The 'Transitive' checkbox is checked.

Domain & Range

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

Classes Object properties Data properties Annotation properties Datatypes Individuals

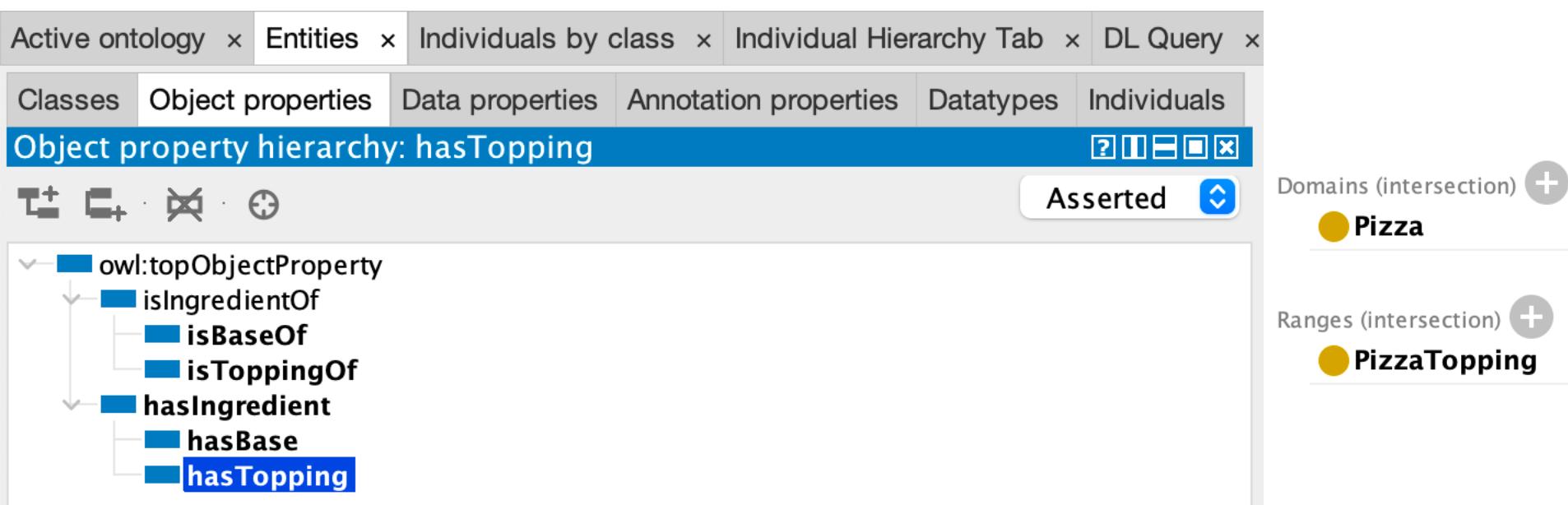
Object property hierarchy: hasTopping

Domains (intersection) +
Pizza

Ranges (intersection) +
PizzaTopping

Asserted

owl:topObjectProperty
 isIngredientOf
 isBaseOf
 isToppingOf
 hasIngredient
 hasBase
 hasTopping



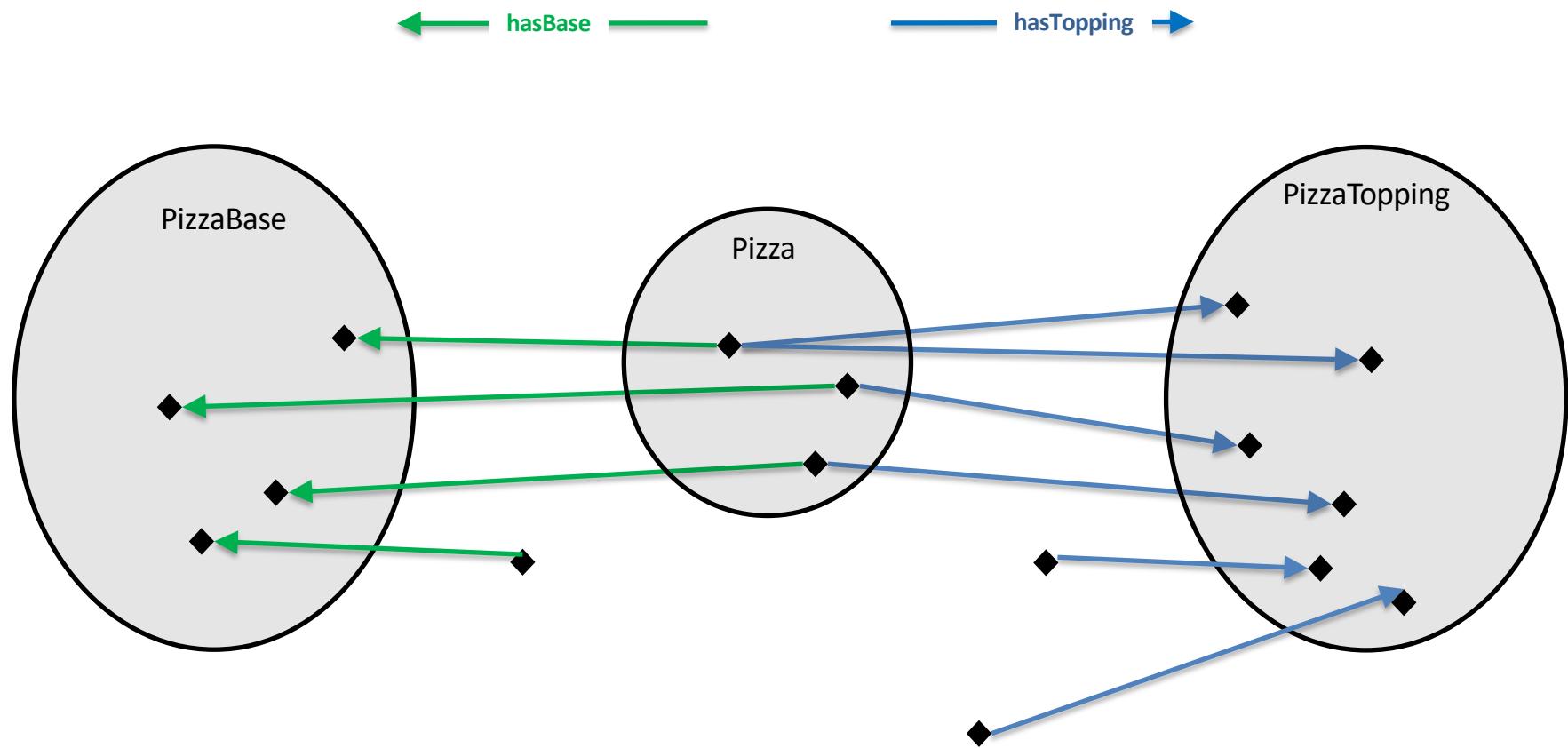
hasBase: Domain Pizza, Range PizzaBase (utiliser autocomplete)



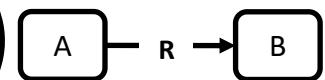
Property Domains And Ranges In OWL — It is important to realise that in OWL domains and ranges should *not* be viewed as constraints to be checked. They are used as ‘axioms’ in reasoning. For example if the property **hasTopping** has the domain set as **Pizza** and we then applied the **hasTopping**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

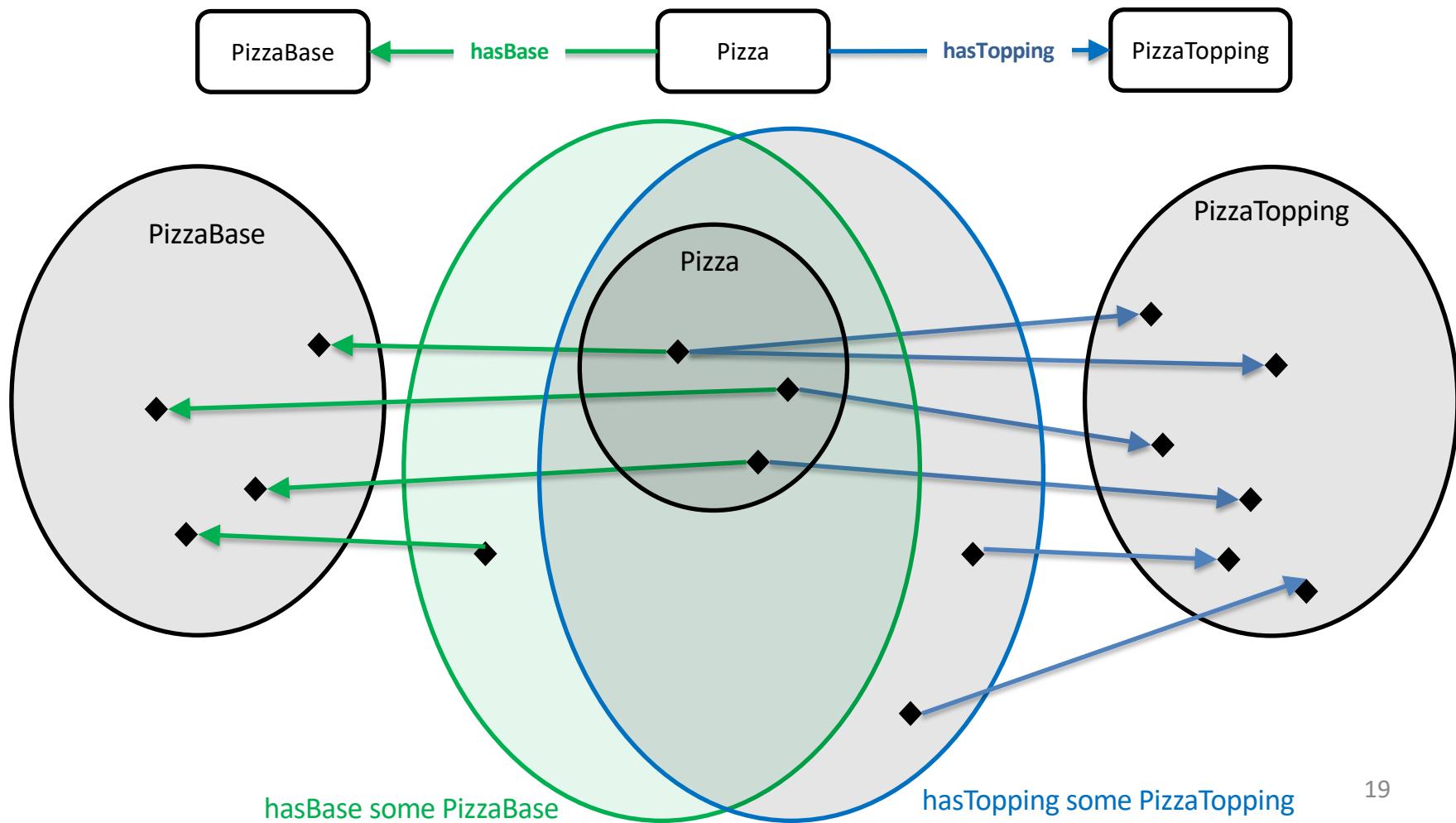
Axiomes existentiels (restrictions existentielles)



Axiomes existentiels (restrictions existentielles)

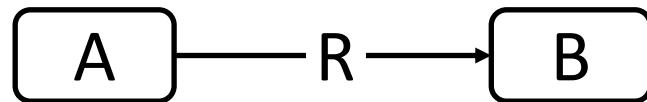


A SubclassOf R some B



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)

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

Raisonneur : détecter les incohérences

Description: ProbInconsistentTopping

Equivalent To +

SubClass Of +

- CheeseTopping
- VegetableTopping

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Class hierarchy: owl:Thing

Asserted

- owl:Thing
 - Pizza
 - NamedPizza
 - AmericanaPizza
 - MargheritaPizza
 - PizzaBase
 - DeepPanBase
 - ThinAndCrispyBase
 - PizzaTopping
 - CheeseTopping
 - ProbInconsistentTopping
 - MozarellaTopping
 - ParmesanTopping
 - MeatTopping
 - VegetableTopping

Reasoner Tools Refactor Wind

Start reasoner ⌘R

Synchronize reasoner

Stop reasoner

Explain inconsistent ontology

Configure...

ELK 0.6.0

FaCT++ 1.6.5

Hermit 1.4.3.456

Mastro DL-Lite Reasoner

Ontop 1.18.1

✓ Pellet

Pellet (Incremental)

jcel

None

Classes Object properties Data properties

Class hierarchy: owl:Thing

Inferred

- owl:Thing
 - owl:Nothing
 - ProbInconsistentTopping
 - Pizza
 - PizzaBase
 - PizzaTopping

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)

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

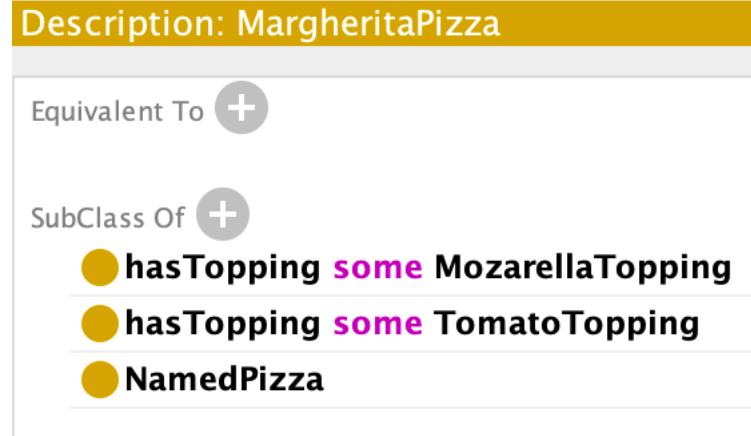
Description: VegetarianPizza

Equivalent To +

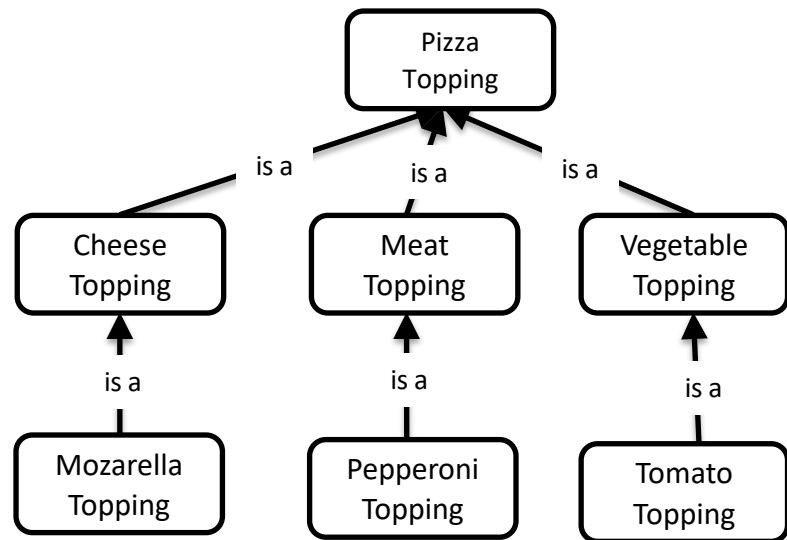
- Pizza
- and (hasTopping only
(CheeseTopping or VegetableTopping))

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 :
 - Une MozarellaTopping pourrait être un MeatTopping
→ nécessité de disjonctions
 - Une MargeritaPizza pourrait avoir un MeatTopping
→ nécessité de restrictions universelles

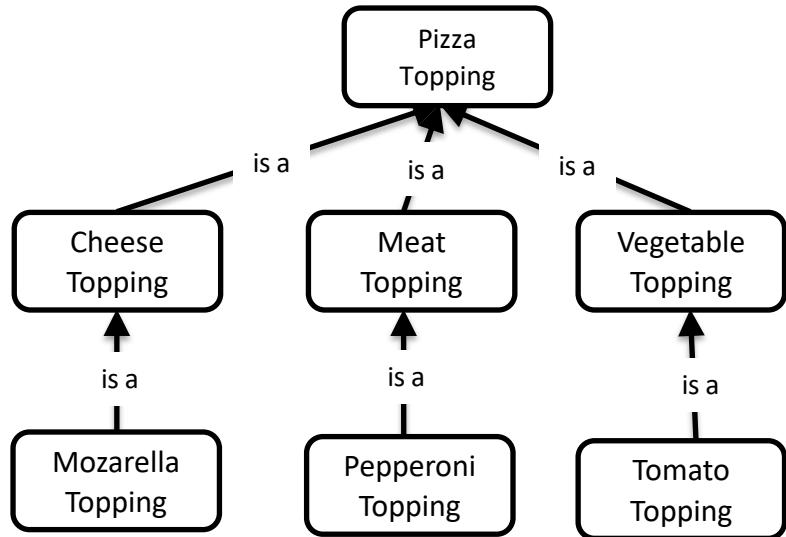


Hypothèse du monde ouvert



Hypothèse du monde ouvert

BD : monde fermé



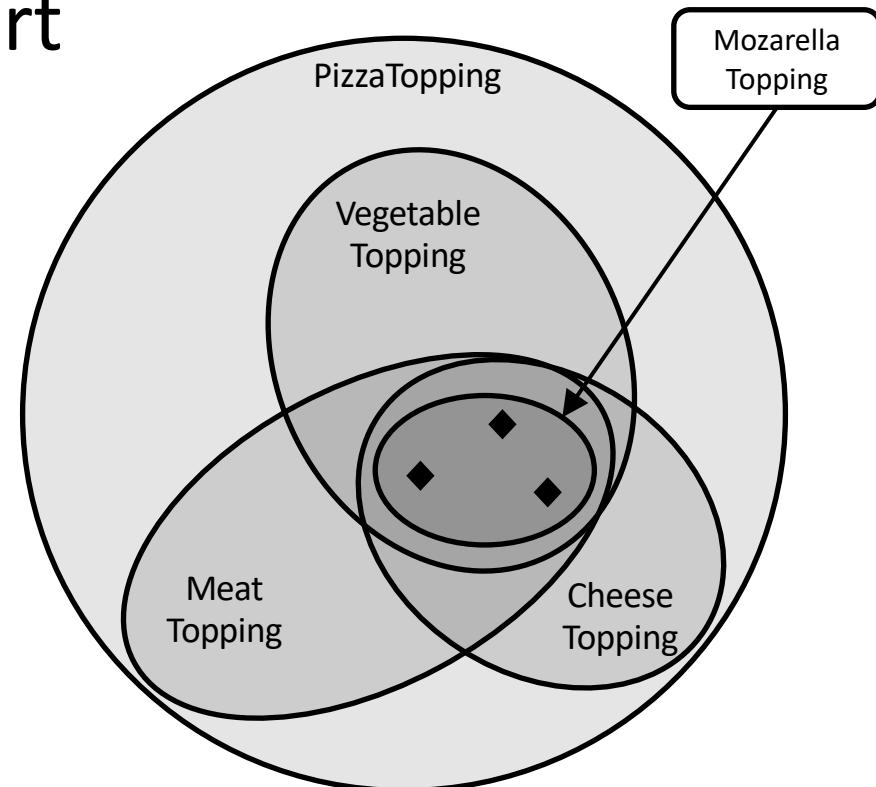
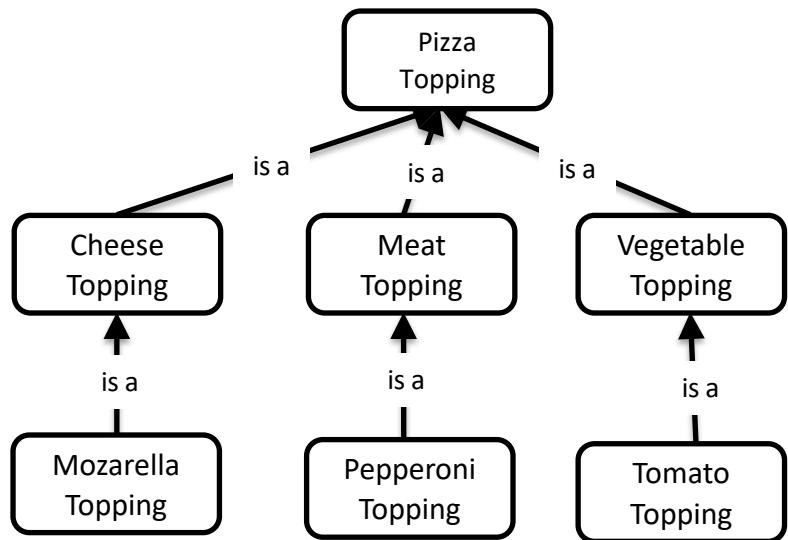
The diagram illustrates a mapping from specific toppings to their respective categories. The top table defines the categories, and the bottom table provides the detailed mappings.

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

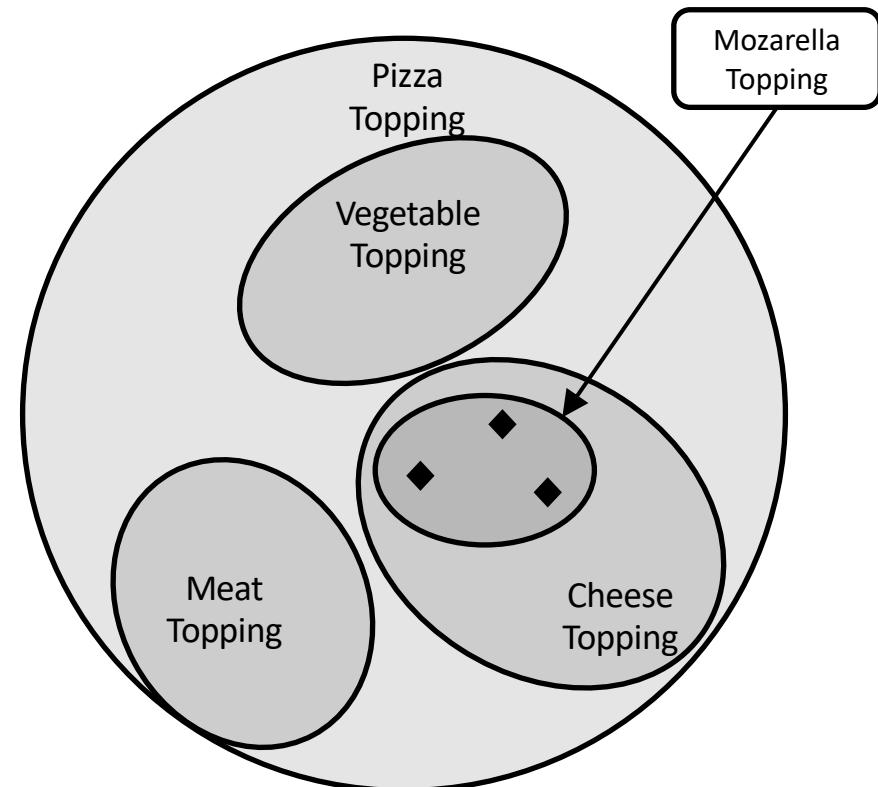
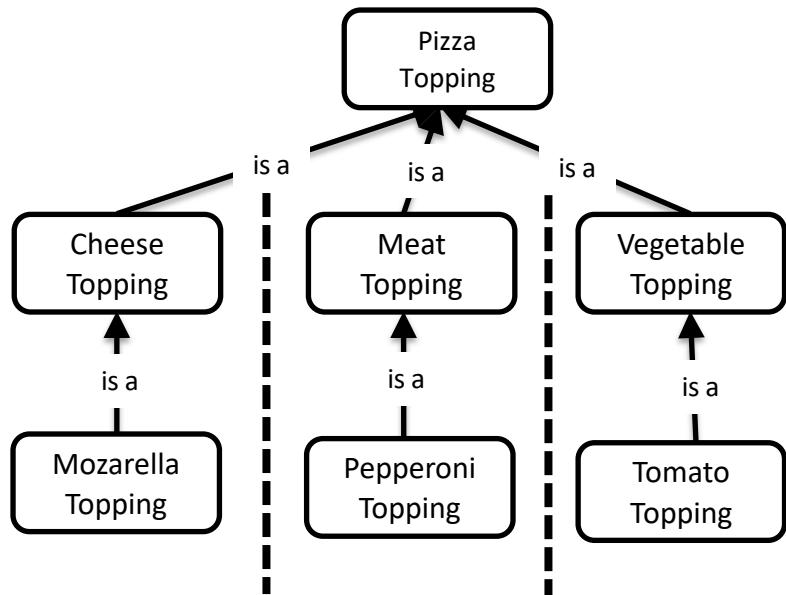
Hypothèse du monde ouvert

Ontologie : monde ouvert



Hypothèse du monde ouvert

Disjonction



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 +

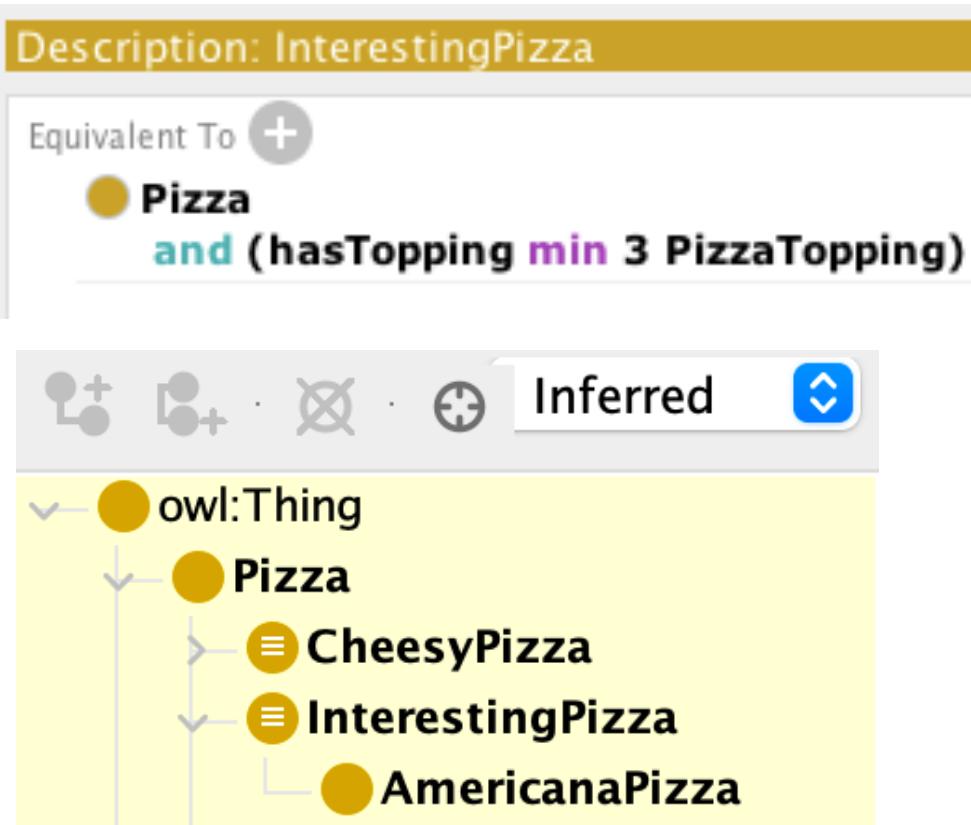
- hasTopping **only** (MozarellaTopping or TomatoTopping)
- hasTopping **some** MozarellaTopping
- hasTopping **some** TomatoTopping
- NamedPizza

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

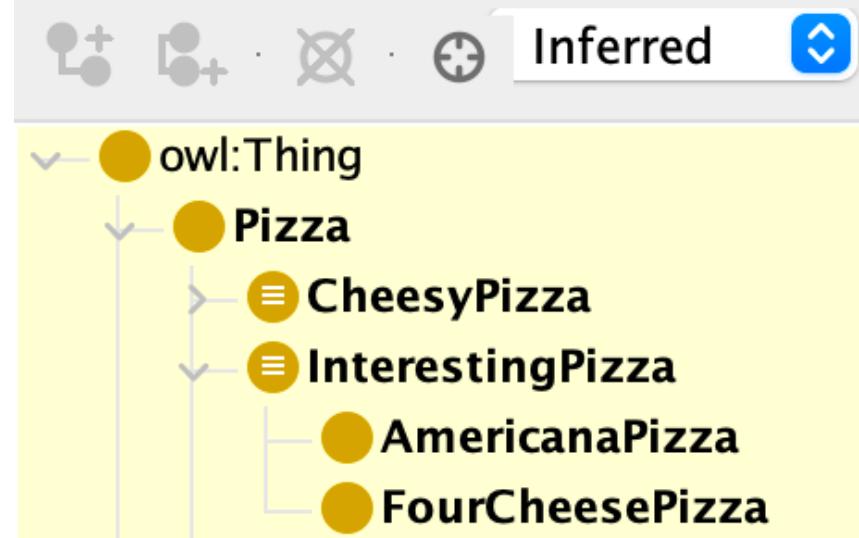


Description: FourCheesePizza

Equivalent To +

SubClass Of +

- hasTopping exactly 4 CheeseTopping
- NamedPizza



Datatype properties et individus

Individuals by type Annotation property hierarchy Datatypes

Object property hierarchy Data property hierarchy

Data property hierarchy: owl:topDataProperty

Asserted

Character Functional

owl:topDataProperty
hasCalorificContentValue

Active ontology Entities Individuals by class

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Individuals: Example-Margherita

Example-Margherita

Value: 263

Language Tag

Datatype: xsd:decimal

Annuler OK

Description: Example-Margherita

Types: MargheritaPizza

Object property assertions

Data property assertions: hasCalorificContentValue 263

Same Individual As

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 Asserted

- owl:topDataProperty
- hasCalorificContentValue

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

Class expression editor

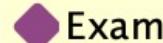
Class hierarchy

Object restriction creator

Data

hasCalorificContentValue **some** xsd:integer[<400]**Description:** LowCaloriePizzaEquivalent To **+****Pizza****and** (hasCalorificContentValue **some** xsd:integer[< 400])SubClass Of **+****Pizza**General class axioms **+**

SubClass Of (Anonymous Ancestor)

**hasCalorificContentValue** **some** xsd:integer**hasBase** **some** PizzaBaseInstances **+****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?

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

Active ontology Entities Individuals by class

Direct instances: Italy ? I B S X

For: ● Country

■ Italy

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Object property hierarchy: hasCoun ? I B S X

+ - X + Asserted ▼

owl:topObjectProperty
hasCountryofOrigin
hasIngredient
isIngredientOf

Description: MozarellaTopping

Equivalent To +

SubClass Of +

● CheeseTopping

● hasCountryofOrigin value Italy

Questions ?

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

Adrien Barton^{1,2}

Avec la collaboration de Paul Fabry²

¹ CNRS, IRIT, Université de Toulouse

² GRIIS, Université de Sherbrooke

4 juin 2024



GRIIS