

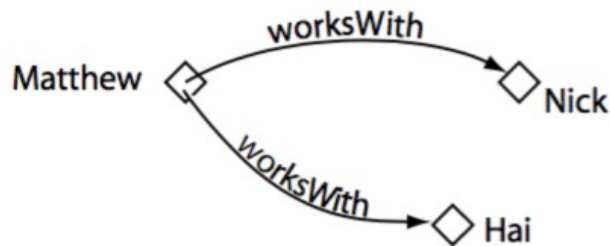
## Prise en main de Protégé (2ème partie)

### Restrictions de cardinalité.

Il s'agit de décrire la classe d'individus qui ont au moins, au plus ou exactement un nombre donné de relations avec d'autres individus ou valeurs.

Attention, deux relations sont considérées distinctes s'il peut être déterminé que les individus des deux relations sont distincts les uns des autres.

Par exemple, sur la figure ci-dessous, l'individu Matthew satisfait une restriction de cardinalité minimum de 2 pour la propriété `worksWith` si les individus Nick et Hai sont des individus différents.



☞ Créer la classe définie `InterestingPizza`, sous-classe de `Pizza`, qui correspond aux pizzas avec au moins 3 garnitures.

☞ Lancer le raisonneur.

Vous devriez trouver `AmericanaPizza`, `AmericanHotPizza` et `SohoPizza` comme sous-classes de `InterestingPizza` dans la hiérarchie inférée.

### Restrictions de cardinalité qualifiées.

Elles sont plus précises que les restrictions de cardinalité, car elles précisent la classe d'objets dans la restriction.

☞ Créer la classe définie `FourCheesePizza`, sous-classe de `NamedPizza`, qui correspond aux pizzas qui ont exactement quatre relations `hasTopping` avec des garnitures `CheeseTopping`. Faites en sorte qu'il ne puisse y avoir aucune garniture d'un autre type.

Nous allons maintenant travailler sur les individus.

### Les individus et les propriétés de types de données

Elles lient un individu à une valeur typée ou non (un littéral RDF ou une valeur d'un type XML Schema).

Les **propriétés de types de données** sont créées à partir de l'onglet `Entities`, puis l'onglet `Data property hierarchy` dans le cadre en bas à gauche.

☞ Créer une propriété `hasCalorificContentValue` qui indique le nombre de calorie d'une pizza.

Nous allons considérer que toutes les pizzas d'un même type (Margherita par exemple) n'ont pas le même nombre de calories, et nous allons donc indiquer ce nombre non pas au niveau des classes

mais au niveau des individus.

Les individus sont créés à partir de l'onglet de la fenêtre Individual by class, dans le cadre Instances.

☞ Créer un individu Example-Margherita : sélectionner la classe MargheritaPizza dans la hiérarchie de classes, puis cliquer sur le bouton Add instance (le choix du type peut également être fait ensuite dans le cadre Description, partie Types).

☞ Ajouter à cette pizza un nombre de calories de 263 de type integer : Data property assertion dans le cadre Property assertions.

☞ Créer une instance de FourCheesePizza avec 723 calories.

☞ Ajouter d'autres individus pizzas.

Il est possible d'imposer que les individus d'une classe aient une propriété de type de données particulière et le type de données de la valeur associée (par exemple toutes les pizzas doivent avoir un nombre de calories) qui est un entier. Ceci est fait avec une restriction.

☞ Dans l'onglet Entities, sélectionner pizza dans la hiérarchie et cliquer sur SubClass Of dans le cadre Description. Dans l'onglet Data restriction editor, saisir hasCalorificContentValue some integer.

☞ Pour que chaque pizza ne puisse avoir qu'un seul nombre de calories associé, préciser que la propriété est fonctionnelle. Tester en ajoutant une deuxième propriété avec 264 calories pour Example-Margherita, votre ontologie devient inconsistante.

Il est également possible de spécialiser un type de données en spécifiant des restrictions sur les valeurs possibles, par exemple un intervalle de valeurs pour un nombre.

☞ Créer une classe HighCaloriePizza, sous-classe de Pizza.

Ajouter une restriction à la description de la classe (SubClass Of) : dans l'onglet Class expression editor taper hasCalorificContentValue some integer[>= 400] (attention à l'espace avant 400). Passer la classe en classe définie.

☞ Créer maintenant une classe LowCaloriePizza pour les pizzas de moins de 400 calories.

☞ Lancer le raisonneur.

Vous devez voir apparaître les instances de ces deux classes dans l'onglet hiérarchie inférée, dans la partie description de la classe.

**Description: HighCaloriePizza**

Equivalent To +

● **Pizza and (hasCalorificContentValue some integer[>= 400])**

SubClass Of +

● Pizza

General class axioms +

SubClass Of (Anonymous Ancestor)

● hasBase some PizzaBase

● hasCalorificContentValue some integer

Instances +

◆ Example-FourChesse

## Restrictions sur la valeur des propriétés

La restriction `hasValue` décrit l'ensemble d'individus qui ont au moins une relation via la propriété spécifiée avec un individu spécifique.

☞ Créer une classe `Country` avec les individus `American`, `England`, `France`, `Italy` et `Germany`.  
Rappel, OWL n'a pas l'hypothèse d'unicité du nom, et on peut donc poser des assertions sur les individus qui peuvent être identiques ou différents d'autres individus.


On va spécifier l'origine des ingrédients de l'ontologie.

☞ Ajouter une nouvelle object property `hasCountryOfOrigin`.  
Ajouter une nouvelle restriction à la classe `MozzarellaTopping` et dans l'onglet `Class expression editor`, taper `hasCountryOfOrigin value Italy`

Attention, le raisonneur ne traite pas complètement la classification des individus actuellement.

## Classes énumérées

Il est possible de définir précisément une classe en listant ses individus membres. Nous allons transformer la classe `Country` en classe énumérée.

☞ Dans le cadre `Description de la classe Country`, dans la partie `Equivalent classes`, cliquer sur  et taper dans l'onglet `Class expression editor` `{America, England, France, Germany, Italy}`.

## Plus sur le raisonnement en monde clos

☞ Créer une pizza `NonVegetarianPizza`, sous-classe de `Pizza`, complémentaire de la classe `VegetarianPizza`.

Pour en faire le complémentaire de la classe `VegetarianPizza`, éditer la restriction `Pizza` du cadre `Description de la classe NonVegetarianPizza` et remplacer par `Pizza and not VegetarianPizza`.

En faire une condition nécessaire et suffisante (classe définie).

☞ Lancer le raisonneur.

Vous devez voir dans la hiérarchie inférée que `FourCheesePizza`, `MargheritaPizza` et `SohoPizza` sont classées comme sous-classes de `VegetarianPizza`, `AmericanaPizza` et `AmericanHotPizza` comme sous-classes de `NonVegetarianPizza`.

Cependant, ajoutons une pizza qui n'a pas l'axiome de fermeture sur la propriété `hasTopping`.

☞ Créer une pizza `UnclosedPizza`, sous-classe de `NamedPizza`, avec une garniture `MozzarellaTopping` (comme on ne spécifie pas l'axiome de fermeture, elle peut donc avoir d'autres garnitures).

☞ Lancer le raisonneur.

On peut constater que `UnclosedPizza` n'a été classé ni dans `VegetarianPizza`, ni dans `NonVegetarianPizza` (ce qu'on pouvait attendre), car en raison de l'absence d'axiome de fermeture, cela ne peut être déterminé.

Egalement dans le tutoriel : les annotations, les ensembles de conditions nécessaires et suffisantes multiples (pages 94 à 97) et des annexes complétant certaines notions.

## Corrigés

- InterestingPizza : pizzas avec au moins 3 garnitures

**Description: InterestingPizza**

Equivalent To

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

SubClass Of

General class axioms

SubClass Of (Anonymous Ancestor)

**hasBase** **some** PizzaBase

- FourCheesePizza : pizzas avec exactement 4 garnitures de type CheeseTopping et aucune autre

**Description: FourCheesePizza**

Equivalent To

SubClass Of

**(hasTopping** **only** CheeseTopping) **and** (hasTopping **exactly** 4 CheeseTopping)  
 **NamedPizza**

General class axioms

SubClass Of (Anonymous Ancestor)

**hasBase** **some** PizzaBase  
 **hasCalorificContentValue** **some** integer

Instances

**Example-FourChesse**