# Cairo University

**Faculty of Computers & Artificial Intelligence**

**Computer Science Department**

**Year 2021 – 2022**

**First Term**

**Lab 3**

**A Protégé Example**

**A Pizza Example:**

1- Create a new Protégé-OWL Project: New Project -> OWL/RDF Files. 2- Create the following classes:

Pizza PizzaBase PizzaTopping

1. Now use multi subclasses creation by right clicking on class to have the following class hierarchy:

Pizza PizzaBase PizzaTopping

CheeseTopping

MozzarellaTopping ParmezanTopping

MeatTopping

PepperoniTopping SalamiTopping

VegetableTopping

OliveTopping TomatoTopping PepperTopping

## Disjoint Classes:

1. An individual can’t belong to Pizza and PizzaBase or PizzaTopping at the same time.

Make all siblings of Pizza as disjoint (mutually disjoint). Also make all Toppings classes disjoint.

1. Create (object) properties: hasIngredient

hasBase hasTopping

hasIngredient: has inverse “isIngredientOf” and both are transitive(a->b, b->c, then a->c) hasBase: is functional -> each Pizza has only one base. It has inverse isBaseOf hasTopping: has inverse “isToppingOf”.

## Domains and Ranges

Specify Domains and Ranges of Properties:

**hasTopping**

Pizza

PizzaTopping

\

Pizza

**hasBase**

PizzaBase

## Existential Quantifiers (∃):

At least *one, or some* quantifier.

Specifying **Existential Quantifiers (**∃ **)** for properties:

A pizza must have a Base: select Pizza, from the conditions widget, create restriction:

### ∃ (Quantifier), hasBase (property), PizzaBase.

1. Create subclass of Pizza

---NamedPizza

--MargheritaPizza(subclass of NamedPizza)

Create the following restrictions on MargheritaPizza:

### ∃ (Quantifier), hasTopping (property), MozzarellaTopping.

∃ **(Quantifier), hasTopping (property), TomatoTopping.**

**Running the Classifier:**

The classifier checks consistency of concepts and creates an inferred hierarchy.

1. Create a class InconsistentTopping as a subclass of CheeseTopping,VegetableTopping. Now InconsistentTopping has 2 superclasses : CheeseTopping and VegetableTopping. Now run the classifier and see what happens. It produces an error on our recently created class because it inherits from 2 disjoint classes.

## Specifying Universal Quantifiers (∀ ) for Closure Axioms:

1. Create class VegetarianPizza as a subclass of Pizza where any pizza is vegetarian if it has a CheeseTopping or a VegetableTopping. No VegetarianPizza can have any toppings other than those two.

### ∃ hasTopping, CheeseTopping.

∃ **hasTopping, VegetableTopping.**

(∀ **hasTopping (CheeseTopping or VegetableTopping) [union operator].**

## Run the reasoner

– expected behaviour:

Margherita pizza should be classified as vegetarian pizzas

– actual behaviour

reasoner does not find any vegetarian pizza subclasses

**OWA (Open World Assumption):**

We can’t assume a piece of information is false if it’s not explicitly stated as false, otherwise it’s just unknown.

1. We can’t assume margherita pizza doesn’t have any other toppings unless explicitly stated.

Now we want to make MargheritaPizza closed on Cheese and Tomato toppings:

### hasTopping, CheeseTopping.

**hasTopping, TomatoTopping.**

**hasTopping (CheeseTopping or TomatoTopping) [union operator].**