Ontology building with Protégé

PART I : Exploration

1. Installation of Protégé version 5.5

Download Protégé from http://protegeproject.github.io/protege/installation/

2. Launch Protégé and import the Pizza ontology

1. Execute the bin file: >sh ./install_protege.bin
2. Import the Pizza ontology (Direct Imports then choose « import an ontology contained in a document located on the web ») (cf. Figure 1)
3. Analyse the class hierarchy, the different options that appear on the boxes on the right of the main window. Try to understand the meaning of the different specifications.
4. Explore the tabs *Properties* and *Individual* and observe how they are used
5. Create some instances of Pizza, PizzaBase and PizzaTopping classes
6. Extend the example by creating a new type of pizza that does not exist. For this purpose,
    - Create a new class representing the new pizza. Express how this new pizza is related to other pizzas using disjoint constraints
    - Create a new kind of topping for your pizza
    - Specify for example that your pizza is of French origin.

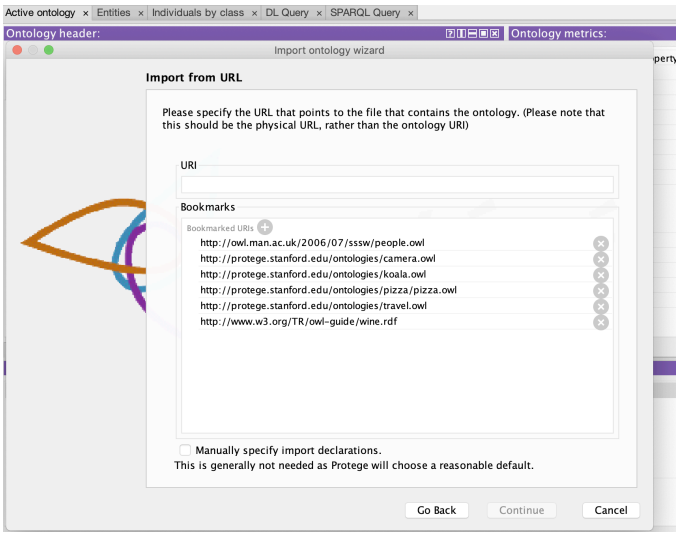

Figure 1 : The Pizza Ontology Import

PART II : Family ontology design and development

The purpose of this *lab* is to give you with the basics of ontology modelling using Protégé tool. The main goal is to design the 'family' ontology, create individuals and infer new relations.

1- Classes and subclasses

The first step is to design classes and subclasses of family ontology according to the following figure (Fig.2):

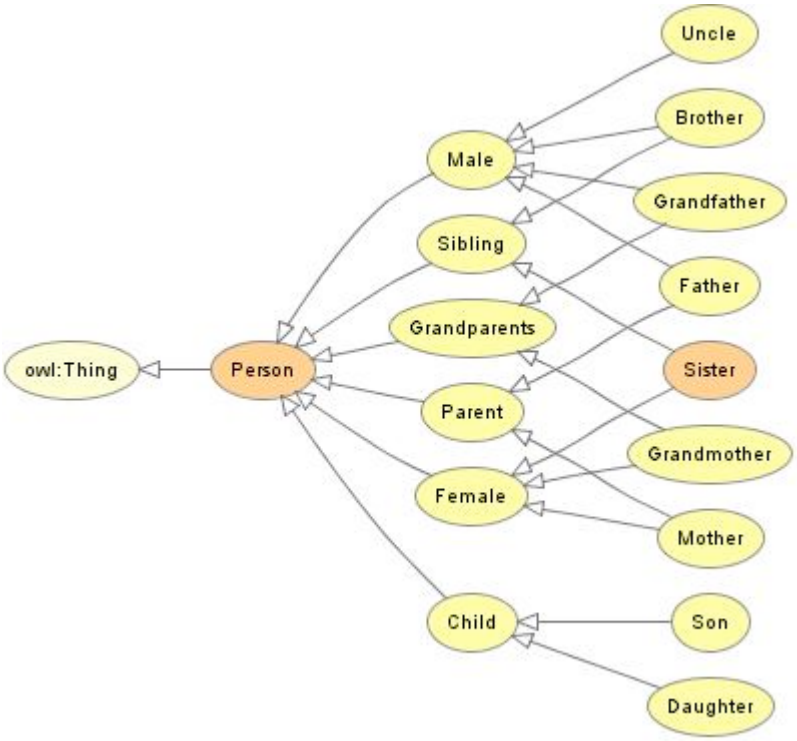

Figure 2 : Family ontology

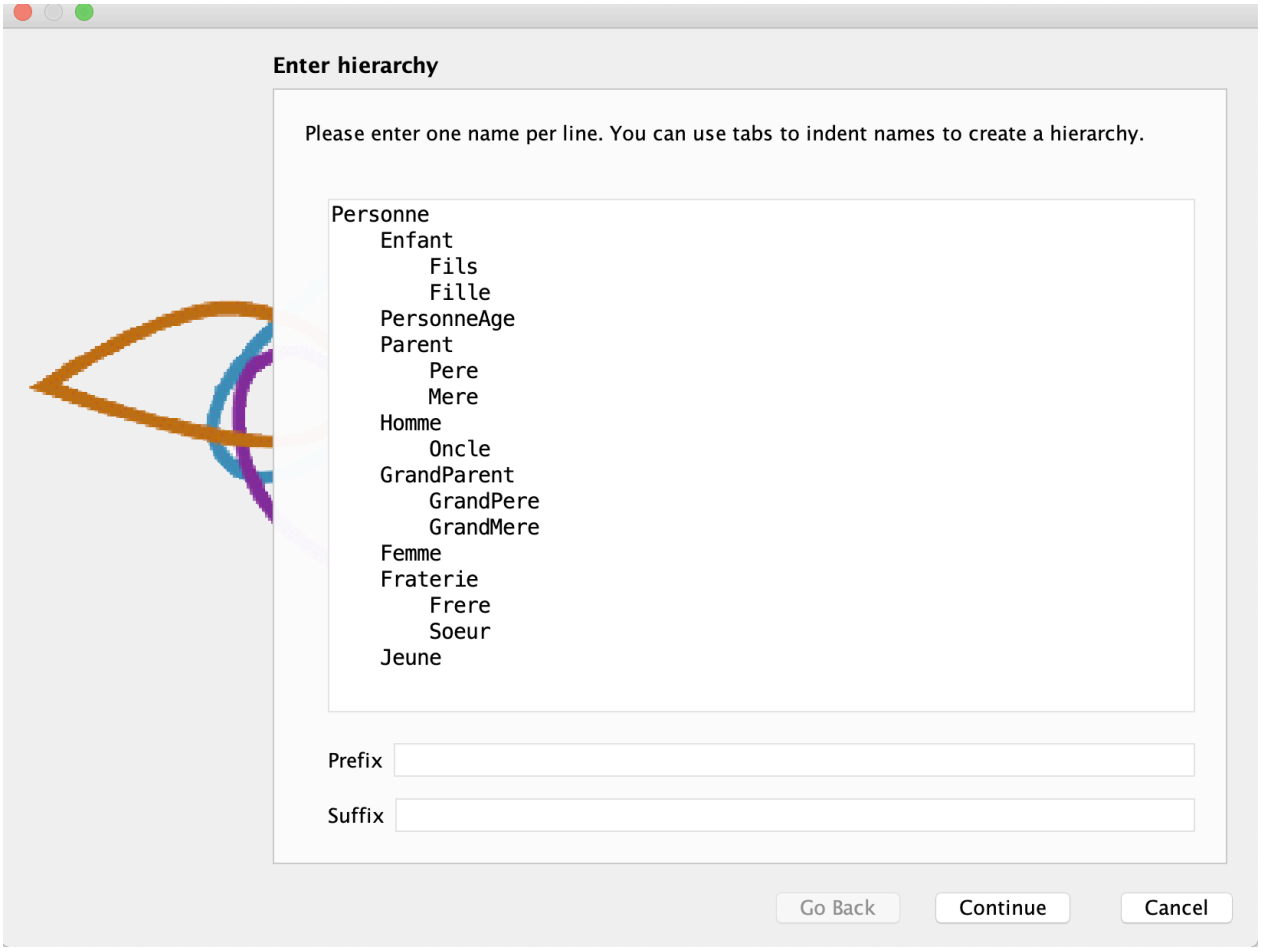

Figure 3 : Sub-classes

Note: It is possible to create a hierarchy of classes and subclasses (with addSubClasses) from a text area using indentations (see Figure 3), or by adding classes one by one with the + button at the top left of the owl:Thing class. Then, to express multiple inheritance, it is sufficient to mention that a class is a subclass of several classes (see Figure 5)

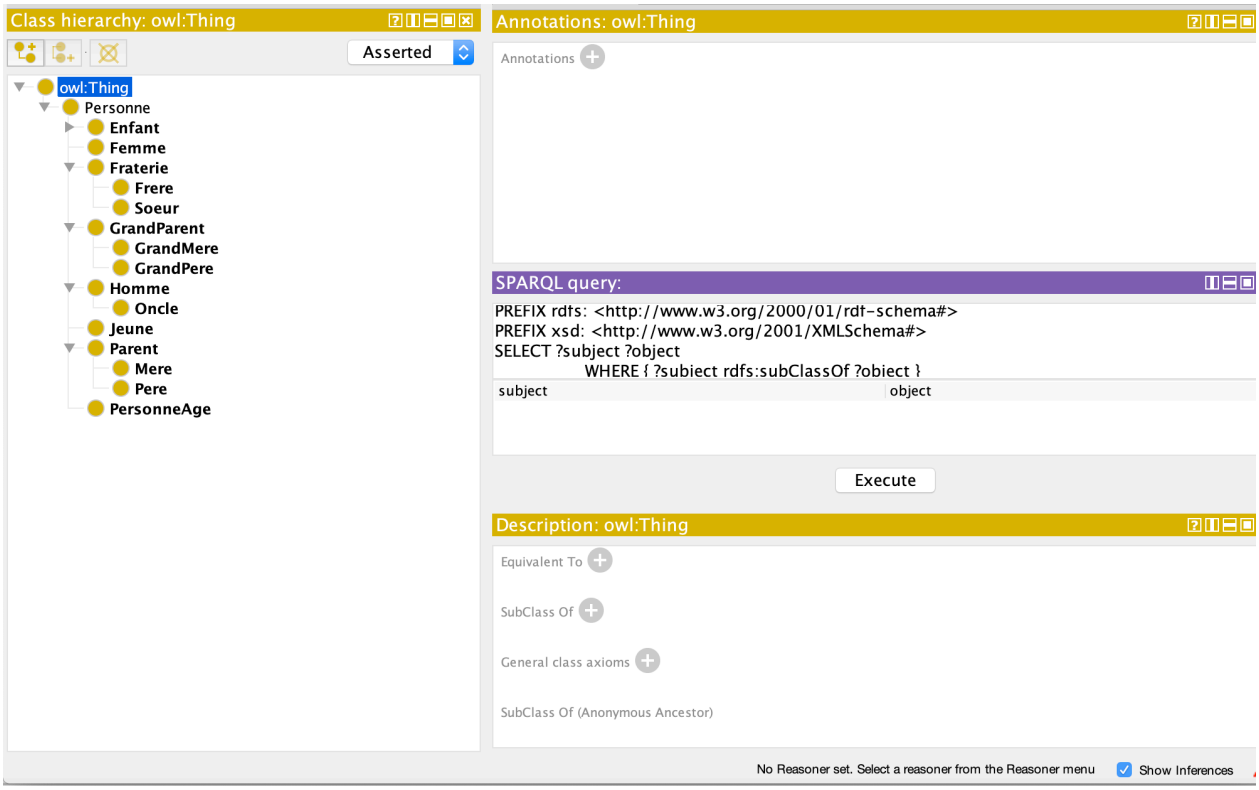

Figure 4 : Family classes

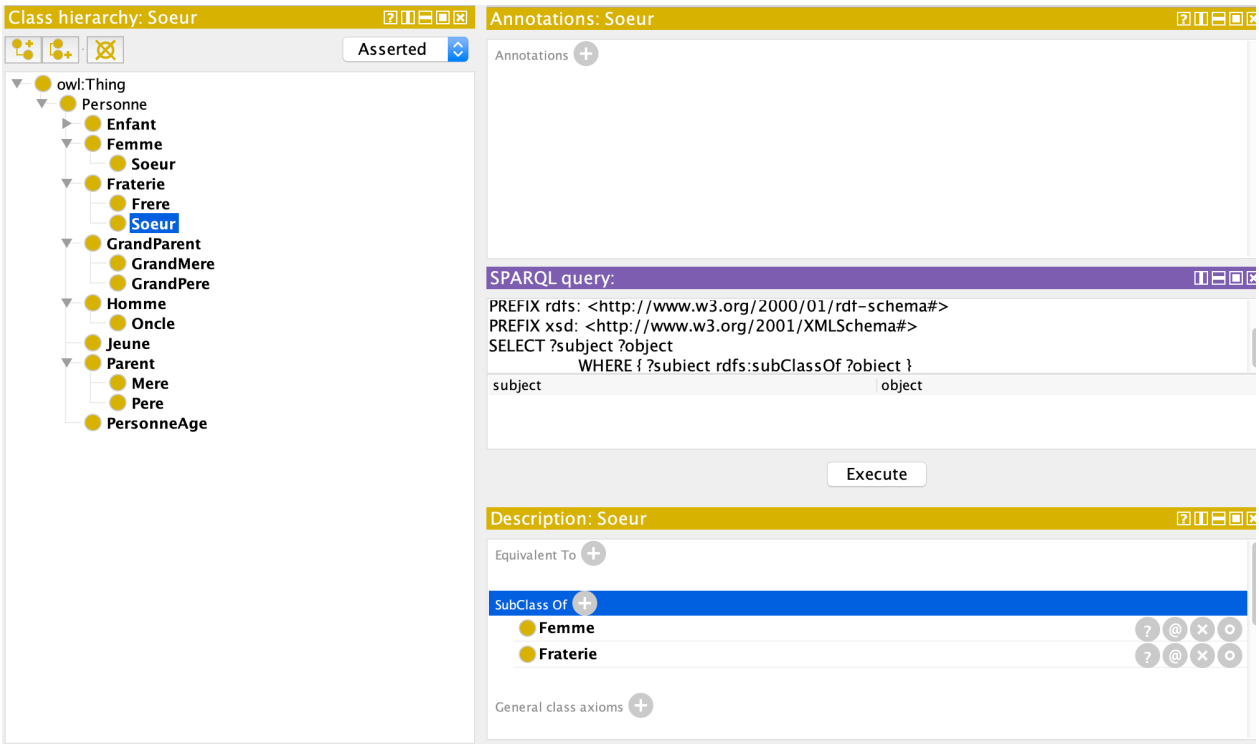

Figure 5 : Multiple Heritage

3. Class properties

a. Data Properties (cf. Figure 6)

1. A person has a name, an age and a nationality.
    a. Create a datatype property *name* with domain **Person** and range xsd:String
    b. Create a datatype property *age* with domain **Person** and range xsd:int
    c. Create a datatype property *nationality* with domain **Person** and range xsd:String

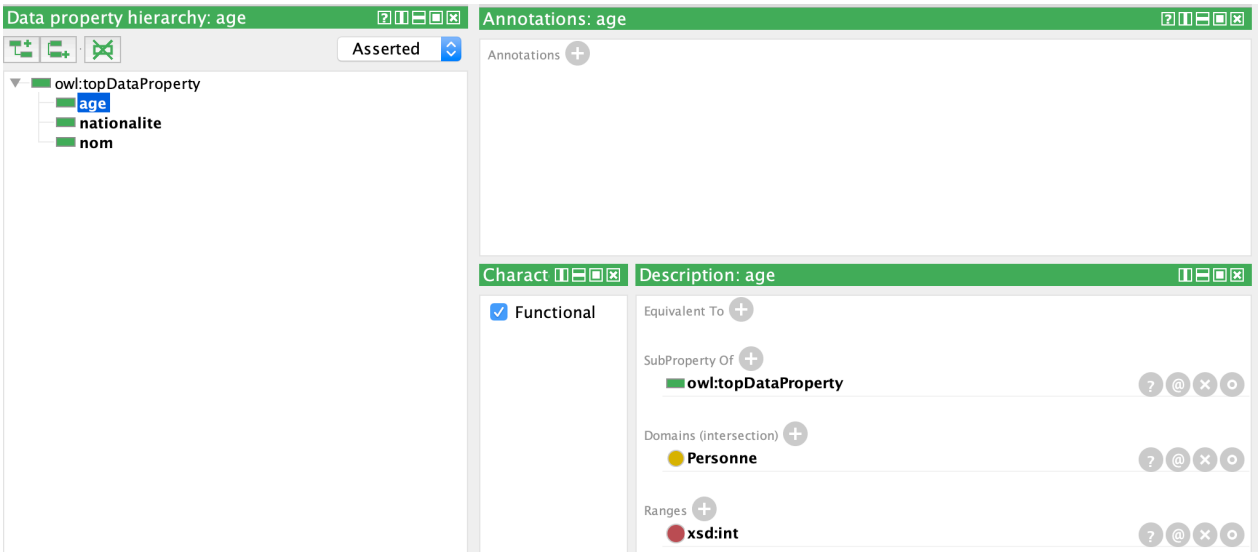

Figure 6 : Data Properties

b. Object Properties (cf. Figure 7)

2. Two people can get married

    Create an object property *isMarriedWith* with **Person** as domain and range
3. A person is parent of another person
    Create the object property isParentOf with **Person** as domain and range

4. A Male is father of person

    Create the object property isFatherOf which is sub property of isParentOf with domain Male and range Person
5. A Female is mother of person
    Create the object property *isMotherOf* which is sub property of *isParentOf* with domain **Female** and range **Person**
6. A person belongs to another person's siblings
    Create the object property *isSiblingOf* with domain **Person** and range **Person**
7. A man is the brother of a person

Create the object property *isBrotherOf* which is sub property of *isSiblingOf* with domain **Male** and range **Person**

8. A Female is the sister of a person
   Create the object property *isSisterOf* which is sub property of *isSiblingOf* with domain **Female** and range **Person**
9. A person is a child of another person
   Create the object property *isChildOf* with domain **Person** and range **Person**
10. A Male is the son of a person
    Create the object property *isSonOf* which is sub property of *isChildOf* with domain **Male** and range **Person**
11. A woman is the daughter of a person

    Create the object property *isDaughterOf* which is sub property of *isChildOf* with domain **Female** and range **Person**

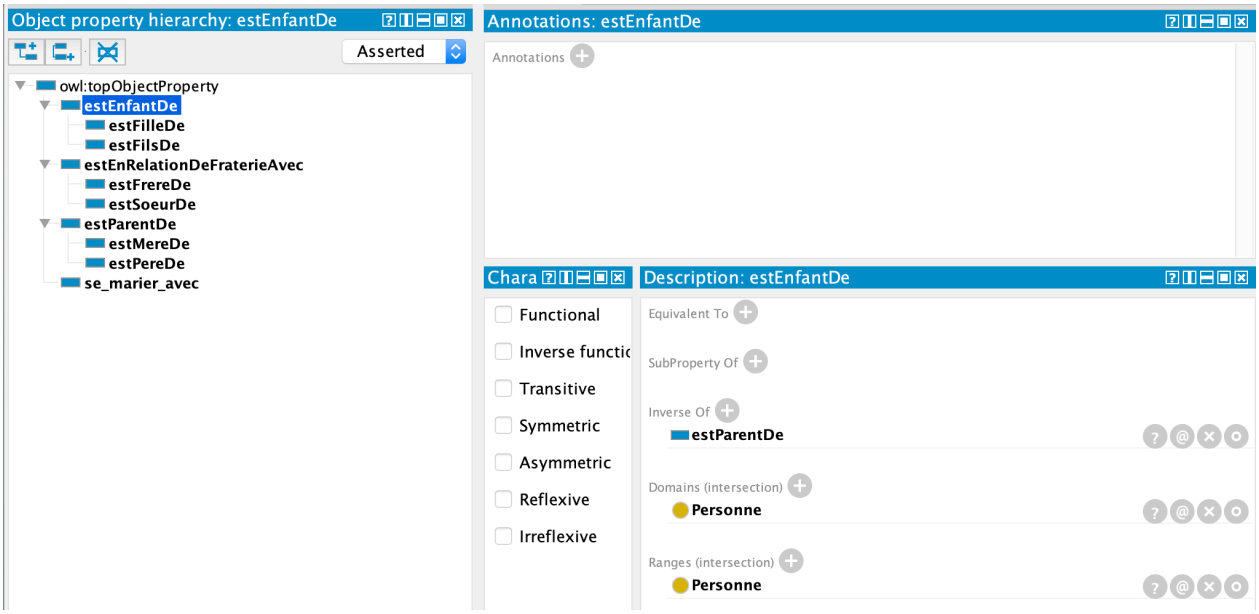

Figure 7 : Object Properties

3. Class and properties restrictions

NECESSARY AND SUFFICIENT CONDITION (Equivalent To):

- An uncle has the restriction : *is brother of one parent*
- A grandfather has the restriction : *is father of a parent*
- A grandmother has the restriction : is mother of a parent
- A father has the restriction : *isFatherOf* property has at least one instance
- A mother has the restriction : *isMotherOf* property has at least one instance
- A son has the restriction : *isSonOf* property has at least one instance

- A daughter has the restriction : *isDaughterOf* property has at least one instance
- A brother has the restriction : *isBrotherOf* property has at least one instance
- A sister has the restriction : *isSisterOf* property has at least one instance (cf. Figure 8)

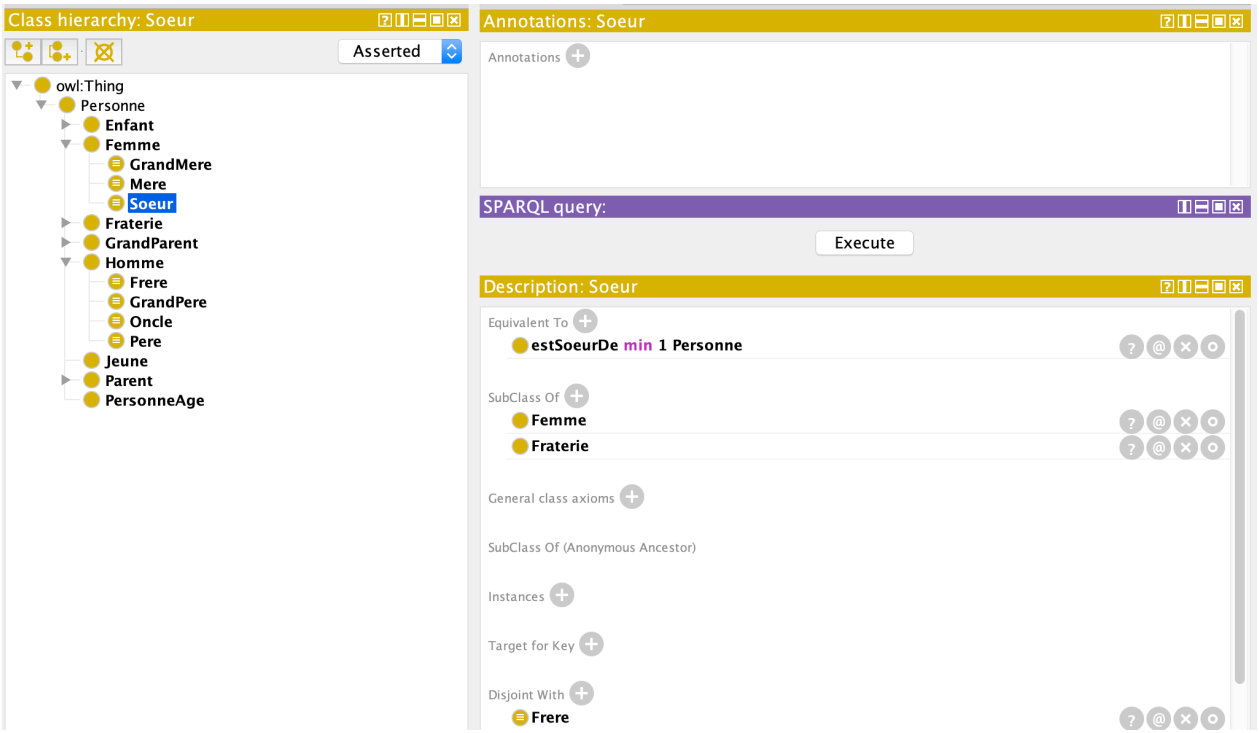

Figure 8 : Restriction on Sœur class

DISJOINTS CLASSES :

- Male and Female are disjoints
- Father and Mother are disjoints
- Son and Daughter are disjoints
- GandFather and GrandMother are disjoints

4. Assign types to properties

1. *iMarriedWith* and *isSiblingOf are* symmetric properties
2. *isSiblingOf* property is transitive
3. *isChildOf* property is the inverse property of *isParentOf*
4. name, age and nationality are functional properties

5. Individuals

1. create individuals to Male class :
    a. Peter, 70, is*MarriedWith* Marie. He is French
    b. Thomas, 40, *isSonOf* Peter. He is French
    c. Paul, 38 *isSonOf* Peter

d. John, 45, is italian
e. Pedro, 10, *isSonOf* John
f. Tom, 10, *isSonOf* Thomas and Alex
g. Michael, 5, *isSonOf* Thomas and Alex

2. create individuals to Female class :
    a. Marie, 69, french
    b. Sylvie, 30, *isDaughterOf* Marie and Peter
    c. Chloé, 18, *isDaughterOf* Marie and Peter
    d. Sylvie is*MarriedWith* John
    e. Claude, 5, *isDaughterOf* Sylvie, french
    f. Alex, 25, is*MarriedWith* Thomas

6. Ontology checking using an inference engine and basic reasoning (OWL)

1. Configure the inference engine, menu : **Reasoner/Pellet or Reasoner/HermiT**
2. Check the consistency by running the reasoner : start reasoner
3. By going through the different instances, you can see the inferences made (yellow background). What do you observe?
4.
5. Instances of the Male and Female classes are automatically associated to the Person class. In version 3 of Protégé these are visible directly from the Person class but since version 4 this is no longer the case. One way to view them is to run a DL Query by typing the class name.
6. No instances are generated in the Brother, Sister and Uncle classes. Why ? How can we infer, for example, the relationships isBrotherOf or isSisterOf from isChildOf?

The next lab (Jena) will allow you to express these kinds of restrictions using a rule language. Indeed, inference rules will solve this problem

SPARQL

1. In the SPARQL Queries window, test some queries
    a. Provide the instances of Male class?
    b. How old is John?
    c. Provide Peter's children names?