Ontology-based modelling and querying

Part 1: Ontology-based modelling

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.1):

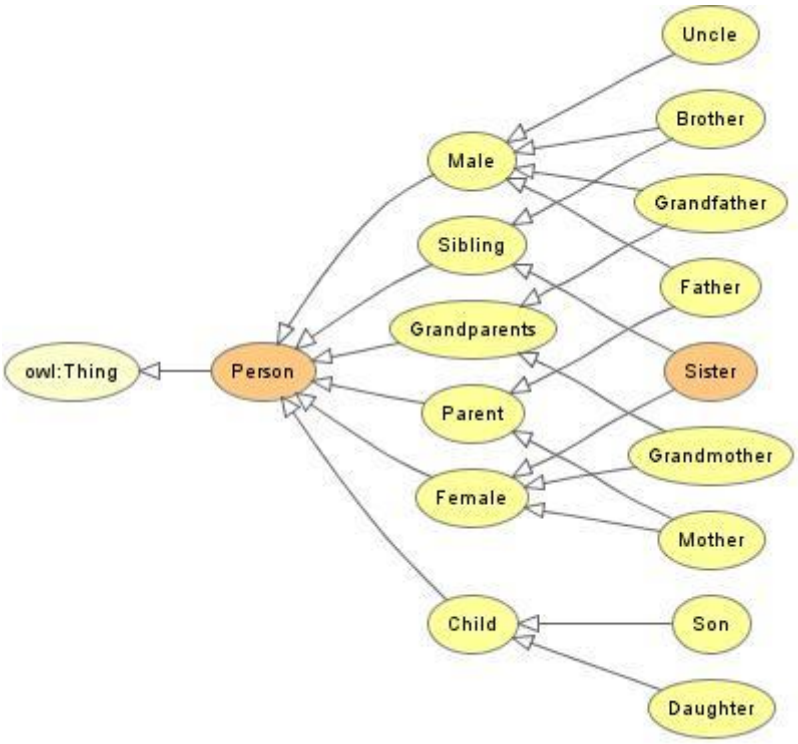

Fig. 1 : Family ontology

Remark: to declare a class C as subclass of two classes A and B, you have to add A and B in asserted condition box (with Logic View)

2. Class properties

1. Datatype properties (Fig. 2): create the properties: name, age and nationality of a person in properties tab which are single properties (functional)

    a. create a datatype property *name* with domain **Person** and range xsd:string
    b. create a datatype property *age* with domain **Person** range xsd:int
    c. create a datatype property *nationality* with domaine **Person** and range xsd:string

2. Object properties (Fig. 3): create the 11 following properties:

    a. Create an object property *isMarriedWith* with **Person as** domain and range

Fig. 2: DataType properties

b. A person is parent of another person

    i. Create the object property *isParentOf* with **Person** as domain and range

c. A Male is father of person

    i. Create the object property *isFatherOf* which is sub property of *isParentOf* with domain **Male** and range **Person**

d. A Female is mother of person

    i. Create the object property *isMotherOf* which is sub property of *isParentOf* with domain **Female** and range **Person**

e. A person belongs to another person's siblings

    i. Create the object property *isSiblingOf* with domain **Person** and range **Person**

f. A man is the brother of a person

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

g. A Female is the sister of a person

    i. Create the object property *isSisterOf* which is sub property of *isSiblingOf* with domain **Female** and range **Person**

h. A person is a child of another person

    i. Create the object property *isChildOf* with domain **Person** and range **Person**

i. A Male is the son of a person

    i. Create the object property *isSonOf* which is sub property of *isChildOf* with domain **Male** and range **Person**

j. A woman is the daughter of a person

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

3. Class and properties restrictions

NECESSARY AND SUFFICIENT CONDITION :

- An uncle has the restriction : *is brother of a 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

DISJOINTS CLASSES: (in Disjoint boxes)

- 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

- *iMarriedWith* and *isSiblingOf* properties *are* symmetric
- *isSiblingOf* property is transitive
- *isChildOf* property is the inverse property of *isParentOf* property
- 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. Chloe, 18, *isDaughterOf* Marie and Peter
    d. Sylvie is*MarriedWith* John

e. Claude, 5, *isDaughterOf* Sylvie, french
f. Alex, 25, is*MarriedWith* Thomas

6. Check ontology consistency with PELLET

1. Check consistency using Pellet

2. Infer instances of the following classes **Person**, **Uncle, Son, Sister, Grandparent, Grandfather, Grandmother;** check the instances generated automatically in the classes Person, Grandparent, Grandfather, Grandmother. No individual is generated in the classes Brother, Sister and Uncle. Why? How to infer for example the relations *isBrotherOf* or *isSister* starting from *childOf*?

In the next Lab (Lab 3), we will see how to express these kinds of restrictions using a rule language. Indeed, inference rules will solve this problem.

Part 2: SPARQL queries

1. Use SnapSparQl plu-ing for this lab

2. Load the family ontology (owl) that you created in part 1.

3. Write the following queries by checking **with OWL inference**:

    1. How old is Peter?
    2. Who are Sylvie's parents?
    3. The women over 30 years?
    4. What are the instances of Person?

Example of a SPARQL query:

```
PREFIX ns: <...#>
SELECT <variable>
WHERE{
    <triplet>.
    <triplet>.
    <fonction>.
}
```

Prefix defines the used namespaces.
? is used to denotes variables (example ?age)

Details about SPARQL can be found at:
http://jena.sourceforge.net/ARQ/Tutorial/

Part 3: query dbpedia.org

Open the public SPARQL endpoint: http://dbpedia.org/snorql/ and test the following query to get information related to the resource http://dbpedia.org/resource/Thomas_Edison

```
SELECT ?property ?hasValue ?isValueOf
WHERE {
 { <http://dbpedia.org/resource/Thomas_Edison> ?property ?hasValue }
 UNION
 { ?isValueOf ?property <http://dbpedia.org/resource/Thomas_Edison> }
}
```

Based on the dbpedia page of Thomas Edison http://dbpedia.org/page/Thomas_Edison, the results of the previous query and using the following public SPARQL endpoint: http://dbpedia.org/snorql/, specify and test the following queries:

1. On what date was Thomas Edison Born?
2. What date did Edison Die?

Other queries:

1. US Presidents born in XXth Century
   Note: use `?President rdf:type <http://dbpedia.org/class/yago/WikicatPresidentsOfTheUnitedStates>.`
2. US Presidents born in XXth Century and if available their death date.
3. List the American recipients of MacArthur Fellowships.