Lab: SPARQL query language and Jena rule language

The aim of this lab is to manipulate both SPARQL query language and Jena rules language. In the first part you will test the queries and the rules by checking the effect of the actions of the latter.

In the next lab, you will be able to program an application with Jena.

SPARQL

- Download the file tpjena_fat.rar and run the application. A mini tutorial describing the application is available on moodle.

1. Checking the constraints defined in your family ontology

Write the following queries by ticking « *With OWL inference* ».

Verify the instances of the classes (they are automatically injected thanks to the constraints defined in OWL).

1. The list (name and age) of Peter's children
2. What are the instances of Person?
3. provide a list of women who are over 30 years old?
4. Display the list of persons (limited to 3) whose father is older than 40
5. Display the list (name and age) of all individuals of French nationality and for each individual, we want the name of his spouse if he is married
6. Display the list of all the persons who are brother of a person
7. Display the list of all persons whose spouse is older
8. List all instances of the class Daughter, for each instance display its name and age if they exist
9. What are the instances of the class Parent with their age if the information exists and display the result in descending order of age

Some basic elements of the syntax of SPARQL:

```
PREFIX ns: <...#>

SELECT <variable> WHERE{

<triplet> . <triplet> . <fonction>.

}
```

Prefix defines the space names.
each variable is written with a? (example: ?age).

To display all the variables used in the result, use select *.
Examples of possible functions:

- FILTER (?price <100) : filter products with a price less than 100 euros.

- OPTIONAL {...} : sets an optional predicate

- UNION

For a detailed tutorial on SPARQL: see

http://jena.sourceforge.net/ARQ/Tutorial/

2. Jena : Writing inference rules to generate new instances

Write the following reasoning rules and to test them, write the associated SPARQL query:

1. If a person A is son of B, then A is an instance of Son.
2. If a person A is a daugther of B, then A is an instance of Daughter.
3. If a person A is son of B, then A is also a child of B
4. If a person A is daughter of B, then A is also a child of B
5. If a person A is a child of B then A is an instance of Child
6. If a person A is a child of B then B is a parent of A
7. If a person A is a mother of B, then A is an instance of Mother
8. If a person A is a father of B, then A is an instance of Father
9. If a person A is a mother of B, then A is also a parent of B
10. If a person A is a father of B, then A is also a parent of B
11. If a person A is a parent of B, then A is an instance of Parent
12. Define the object property isUncleOf as a brother of a parent
13. If a person A has parents who have nationality X, then A has nationality X
14. A person A who is older than 60 is aged (instance of Old)

General structure of the jena rules:
@prefix ns: `<http://www.owl-ontologies.com/Ontology1291196007.owl#>.`

[rulename: (<triplet>) -> (<triplet>)]

Some usefull functions: le(?x,?y), ge(?x,?y), lessThan(?x,?y), greaterThan(?x,?y)

For more details: http://jena.sourceforge.net/inference/#rules