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

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tg:<http://www.turnguard.com/functions#>
SELECT ?peter ?enfantnom ?enfantage
WHERE {
?peter rdf:type ns:Person .
?enfant ns:isChildOf
?enfant ns:name ?enfantnom .
?enfant ns:age ?enfantage .
 }
```

2. What are the instances of Person?

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tg:<http://www.turnguard.com/functions#>
SELECT ?per
WHERE {
?per rdf:type ns:Person . }
```

3. Provide a list of women who are over 30 years old?

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```

```
SELECT ?y
WHERE {
?y rdf:type ns:Female.
?y ns:age ?age.
FILTER(?age>30).
}
```

4. Display the list of persons (limited to 3) whose father is older than 40

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tg:<http://www.turnguard.com/functions#>
SELECT ?personne ?pere ?pereage
WHERE{
?personne rdf:type ns:Person .
?personne ns:isChildOf ?pere.
?pere rdf:type ns:Male .
?pere ns:age ?pereage .
FILTER (?pereage >40 ) }
ORDER BY ( ?pereage) LIMIT 3
```

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

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tg:<http://www.turnguard.com/functions#>
SELECT ?nom ?age ?epousenom ?epouseage
WHERE{
?francais rdf:type ns:Person .
?francais ns:nationality "French".
?francais ns:age ?age.
?francais ns:name ?nom.
OPTIONAL {
?francais ns:isMarriedWih ?epouse .
?epouse ns:name ?epousenom .
?epouse ns:age ?epouseage .
}
}
ORDER BY DESC (?age)
```

6. Display the list of all the persons who are brother of a person

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#> PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX
tg:<http://www.turnguard.com/functions#>
SELECT DISTINCT ?nom ?frerenom
WHERE{
?personne rdf:type ns:Person .
?personne ns:name ?nom .
?personne ns:isChildOf ?parent.
?frere ns:isChildOf ?parent.
?frere rdf:type ns:Male.
?frere ns:name ?frerenom
FILTER (?nom != ?frerenom)

     }
```

7. Display the list of all persons whose spouse is older

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tg:<http://www.turnguard.com/functions#>
SELECT DISTINCT ?nom ?age ?epousenom ?epouseage
WHERE{
?personne rdf:type ns:Person .
?personne ns:name ?nom .
?personne ns:age ?age .
?personne ns: isMarriedWith ?epouse .
?epouse ns:name ?epousenom .
?epouse ns:age ?epouseage .
FILTER (?epouseage > ?age) }
```

8. List all instances of the class Daughter, for each instance display its name and age if they exist

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tg:<http://www.turnguard.com/functions#>
SELECT ?fille ?nom ?age
WHERE {
?fille rdf:type ns:Daughter .
OPTIONAL {
?fille ns:age ?age .
?fille ns:name ?nom . }
}
```

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

```
PREFIX ns: <http://www.it-sudparis.eu/family#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tg:<http://www.turnguard.com/functions#>
SELECT ?parent ?age
WHERE {
?parent rdf:type ns:Parent .
OPTIONAL {?parent ns:age ?age .} }
ORDER BY DESC ( ?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

```
@prefix ns: <http://www.it-sudparis.eu/family#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
[rule1: (?I rdf:type ?A) (?A rdfs:subClassOf ?B) -> (?I rdf:type ?B)]
[rule2: (?A rdfs:subClassOf ?B) (?B rdfs:subClassOf ?C) -> (?A rdfs:subClassOf ?C)]
[rule3_4:(?A rdf:type ns:Person) (?A ns:isSonOf ?B) -> (?A rdf:type ns:Son) (?A ns:isChilOf
?B)]
[rule5_6: (?A rdf:type ns:Person) (?A ns:isDaughterOf ?B) -> (?A rdf:type ns:Daughter) (?A
ns:estEnfantDe ?B)]
[rule7_8_13: (?A ns:estEnfantDe ?B) -> (?A rdf:type ns:Enfant) (?B ns:estParentDe ?A)(?B
rdf:type ns:Parent)]
[rule9_11: (?A rdf:type ns:Personne) (?A ns:estMereDe ?B) -> (?A rdf:type ns:Mere) (?B
ns:estEnfantDe ?A) (?A ns:estParentDe ?B)]
[rule10_12: (?A rdf:type ns:Personne) (?A ns:estPereDe ?B) -> (?A rdf:type ns:Pere) (?B
ns:estEnfantDe ?A) (?A ns:estParentDe ?B)]
[rule13: (?A ns:estParentDe ?B) -> (?A rdf:type ns:Parent)]
[rule14: (?A rdf:type ns:Homme) (?A ns:estParentDe ?B) (?B rdf:type ns:Parent) -> (?A rdf:type
ns:GrandPere)]
[rule15: (?A rdf:type ns:Femme) (?A ns:estParentDe ?B) (?B rdf:type ns:Parent) -> (?A rdf:type
ns:GrandMere)]
[rule16: (?A ns:estParentDe ?B) (?B rdf:type ns:Parent) -> (?A rdf:type ns:GrandParent)]
[rule17: (?A ns:estFrereDe ?B) (?B rdf:type ns:Parent) -> (?A rdf:type ns:Oncle)] [rule18: (?A
ns:estEnfantDe ?P) (?B ns:estEnfantDe ?P) notEqual(?A, ?B) (?A rdf:type ns:Homme)-> (?A
ns:estFrereDe ?B) (?A rdf:type ns:Frere)]
[rule19: (?A ns:estEnfantDe ?P) (?B ns:estEnfantDe ?P) notEqual(?A, ?B) (?A rdf:type
ns:Femme)-> (?A ns:estSoeurDe ?B) (?A rdf:type ns:Soeur)]
[rule20: (?A ns:estEnfantDe ?P1) (?A ns:estEnfantDe ?P2) notEqual(?P1, ?P2) (?P1
ns:nationalite ?X) (?P2 ns:nationalite ?X) -> (?A ns:nationalite ?X)]
[rule21: (?per rdf:type ns:Personne) (?per ns:age ?age) greaterThan(?age, 60) -> (?per
rdf:type ns:PersonneAge)]
[rule22: (?per rdf:type ns:Personne) (?per ns:age ?age) greaterThan(?age, 9) lessThan(?age,
27) -> (?per rdf:type ns:Jeune)]
```