M2 – Data Science

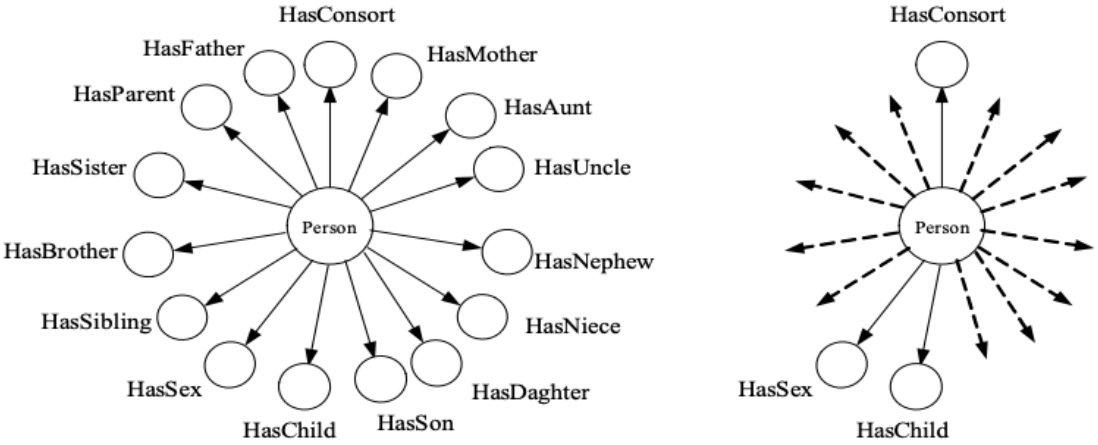
Paris Saclay University

Semantic Web and Ontologies
F. Saïs (sais@lri.fr)

Practical session: Reasoning on Ontologies

1- Family ontology and SWRL rules

The ontology family.owl is an OWL ontology describing the usual classes, e.g., Person, Man, Woman, Child, Daughter, Parent, Father, etc., and relationships, e.g., hasConsort, hasGender, hasChild etc. within a family. Initially, only the hasConsort, hasChild, hasGender properties were filled for men, and hasGender for women



- Figure 1- Person relationships (owl:objectProerties) -

A set of class expressions and equivalences have been defined

1- Person := Man ∪ Woman
2- Parent := Person ∩ hasChild ≥ 1
3- Child := Person ∩ hasParent ≥ 1
4- (hasChild) -1 = hasParent
5- Father := Parent ∩ Man
6- Mother := Parent ∩ Woman
7- Son := Child ∩ Man
8- Daughter := Child ∩ Woman
9- Brother := Sibling ∩ Man
10- Sister := Sibling ∩ Woman
11- Nephew := Man ∩ (hasUncle ≥ 1 ∪ hasAunt ≥ 1)
12- Relative := Child ∪ Parent ∪Aunt ∪ Nephew ∪ Niece ∪ Uncle ∪ Sibling

A set of SWRL rules have also been specified to express additional person relationship or for mirroring the ontology knowledge:

R1- isSiblingOf(?x1,?x2) ^ Man(?x2) -> hasBrother(?x1,?x2)
R2- hasParent(?x1,?x2) ^ Man(?x2) -> hasFather(?x1,?x2)
R3- hasChild(?x1,?x2) ^ Man(?x2) -> hasSon(?x1,?x2)
R4- isPartnerIn(?x2,?x3) ^ hasParent(?x1,?x2) -> hasParent(?x1,?x3)
R5- isSiblingOf(?x1,?x2) ^ hasDaughter(?x1, ?x3) -> isNieceOf(?x3,?x2)
R6- hasChild(?x1,?x2) ^ Woman(?x1) -> hasDaughter(?x1,?x2)
R7- hasChild(?x1,?x2) ^ hasChild(?x3, ?x2) differentFrom(?x1, ?x3) -> isSiblingOf(?x1,?x3)
R8- isSiblingOf(?x1,?x2) ^ hasSon(?x2,?x3) -> isNephewOf(?x3,?x1)
R9- hasParent(?x1,?x2) ^ hasBrother(?x2,?x3) -> hasUncle(?x1,?x3)
R10- hasParent(?x1,?x2) ^ Woman(?x2) -> hasMother(?x1,?x2)
R11- isSiblingOf(?x1,?x2) ^ Woman(?x2) -> hasSister(?x1,?x2)
R12- hasParent(?x1,?x2) ^ hasSister(?x2,?x3) -> hasAunt(?x1,?x3)

To practice:

Open the attached **family.owl** file and use the SWRL Tab to edit the previous SWRL rules.

To see the inferences you can run Drools (/!\ it takes too much time)

2- Reasoning on RDFS and OWL ontology (Home Work to be marked)

A RDFS schema consists of a set of classes (unary relations) organized in a taxonomy and a set of typed properties (binary relations).
These properties can also be organized in a taxonomy of properties. Two kinds of properties can be distinguished in RDFS: the so-called relations, the domain and the range of which are classes and the so-called attributes, the domain of which is a class and the range of which is a set of basic values (e.g. Integer, date, String).

This distinction is also made in OWL which allows to declare Object and datatype properties. For example, in the RDFS schema presented in Figure 2 and corresponding to a cultural application, we have as relation (objectProperty) *Located* having as domain the class *Museum* and as range the class *City*. We also have an attribute *MuseumName* having as domain the class Museum and as range the data type Literal.
Note that, relations and attributes can be renamed in order to ensure that every attribute and every relation has only one domain and one range.

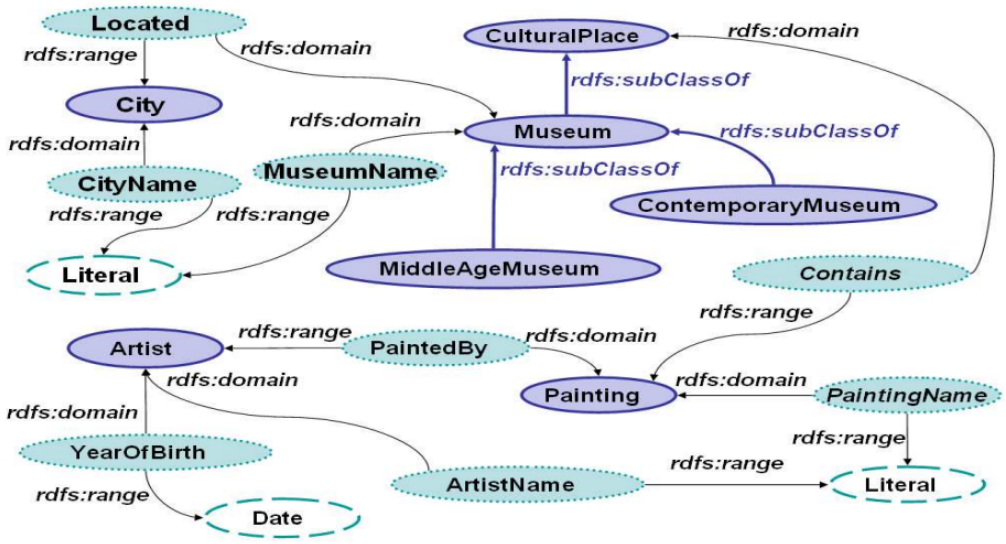


Figure 2- Cultural Ontology (RDFS part)

This schema could be extracted from the following relational schema:

```
CulturalPlace(IDCulturalPlace),
Museum(IDCulturalPlace#, MuseumName, IDCity#, Category),
Painting(IDPainting,PaintingName, IdArtist#, IDCulturalPlace#),
Artist(IDArtist, ArtistName, YearOfBirth), City(IDCity, CityName).
```

Let consider the following RDF triples as described by the cultural ontology and presented in a logical notation (i.e. using unary and binary relations):

```
MuseumName(m1,“musee du LOUVRE”);
Located(m1,c1);
Contains(m1,p1);
Contains(m1, p2);
CityName(c1, “Ville de paris”);
PaintingName(p1, “Joconde’);
PaintingName(p2, “Abricotiers en fleurs”);


MuseumName(m2,“LE LOUVRE”);
Contains(m2,p2);
Located(m2,c2);
CityName(c2,”Paris”);
PaintingName(p3,“La Joconde”);
```

Question 1:

If you apply the RDFS inference rules shown in course 3 (Semantics and Reasoning for Semantic Web), what is the set of inferred facts that you should obtain?

Question 2:

Consider the following OWL axioms as added to the Cultural ontology (in its version of question1) and the inferred facts.

```
owl :disjointWith (CulturalPlace, Artist).
owl :disjointWith (CulturalPlace, City).
owl :disjointWith (CulturalPlace, Painiting).

owl :disjointWith (City, Artist).
owl :disjointWith (City, Painting).
```

If you apply the inference rules expressing the semantics of **owl:disjointWith** shown in course 3 (Semantics and Reasoning for Semantic Web), what is the set of inferred facts that you would obtain?

Question 3:

Consider the following OWL axioms as added to the Cultural ontology (in its version of question2) and the inferred facts.

```
owl:functionalProperty(Located)
owl:functionalProperty(MuseumName)
owl:functionalProperty(CityName)
owl:functionalProperty(PaintingName)
owl:inversefunctionalProperty(contains)
owl:inversefunctionalProperty(MuseumName)
```

Consider also the following SWRL rules added to the Cultural ontology
R1: MuseumName (?x, ?y) ^ MuseumName (?z, ?w) ^(?y = ?w) => owl:sameAs(?x, ?z)
R2: PaintingName (?x, ?y) ^ PaintingName (?z, ?w) ^(?y = ?w) => owl:sameAs(?x, ?z)

And the fact also that: (`'Joconde' = ' La Joconde')`

If you apply the SWRL rules R1 and R2; and inference rules expressing the semantics of `owl:functionalProperty` and `owl:inverseFunctionalProperty` shown in course 3 (Semantics and Reasoning for Semantic Web), what is the set of inferred facts that you would obtain?