Université de Bouira
Département d'Informatique
Module : Ontologies et Web Sémantique
Niveau M2-ISIL

Série TD 5

Les ontologies. Raisonner avec OWL et SWRL.

Exercice 1

1. Exprimer les règles suivantes en codes SWRL XML:

   - Person(?p) ^ hasNumber(?p, ?number) ^ swrlb:startsWith(?number, "+") → hasInternationalNumber(?p, true)

   - Person (?x) ^ (hasChild >= 1)(?x) → Parent(?x)

   - Person(Lambda) ^ hasSibling(Lambda, ?s) ^ Man(?s) → hasBrother(Lambda, ?s)

2. Exprimer les codes SWRL XML suivants sous forme de règles de clauses de Horn:

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#code1"/>
  <ruleml:_body>
    <swrlx:classAtom>
      <owlx:Class owlx:name="&ulan;Artist" />
      <ruleml:var>x</ruleml:var>
    </swrlx:classAtom>
    <swrlx:classAtom>
      <owlx:Class owlx:name="&aat;Style" />
      <ruleml:var>y</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom  swrlx:property="&aatulan;artistStyle">
      <ruleml:var>x</ruleml:var>
      <ruleml:var>y</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="&vra;creator">
      <ruleml:var>x</ruleml:var>
      <ruleml:var>z</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="&vra;style/period">
      <ruleml:var>z</ruleml:var>
      <ruleml:var>y</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#code2"/>
  <owlx:Annotation>
    <owlx:Documentation>Gold customers get a 10% discount on purchases
                        of $500 or more</owlx:Documentation>
  </owlx:Annotation>
  <ruleml:_body>
    <swrlx:individualPropertyAtom  swrlx:property="&ex;#hasStatus">
      <ruleml:var>customer</ruleml:var>
      <owlx:Individual owlx:name="&ex;#gold"/>
    </swrlx:individualPropertyAtom>
    <swrlx:datavaluedPropertyAtom  swrlx:property="&ex;#hasTotalPurchase">
      <ruleml:var>customer</ruleml:var>
      <ruleml:var>total</ruleml:var>
    </swrlx:datavaluedPropertyAtom>
    <swrlx:builtinAtom  swrlx:builtin="&swrlb;#greaterThanOrEqual">
      <ruleml:var>total</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;#int">500</owlx:DataValue>
    </swrlx:builtinAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:datavaluedPropertyAtom  swrlx:property="&ex;#hasDiscount">
      <ruleml:var>customer</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;#int">10</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

Exercice 2 - Lab

I. Installation et configuration de Protégé

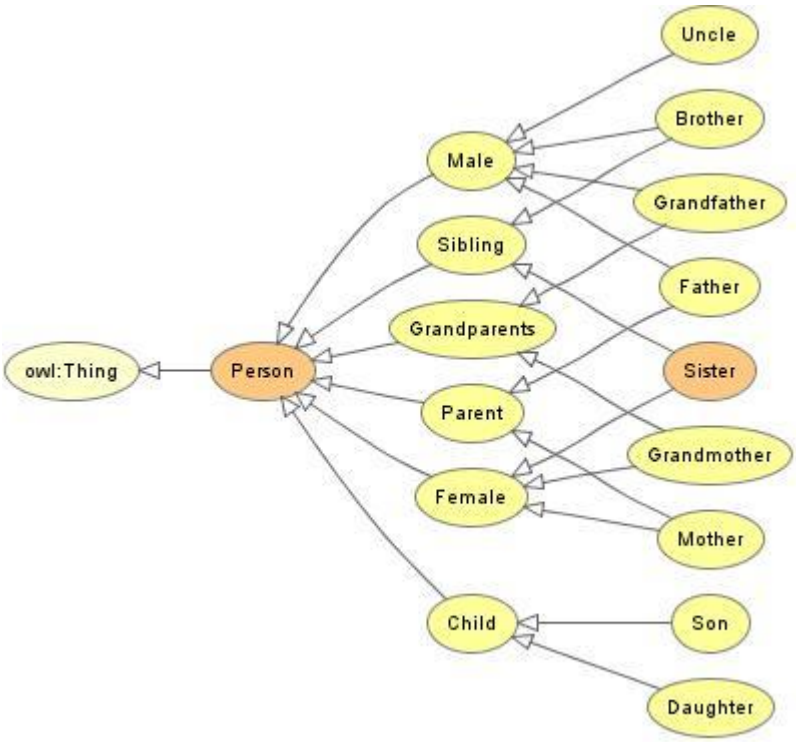
1. Télécharger la version desktop platform-independant de Protégé 5.0 depuis : *https://protege.stanford.edu/products.php#desktop-protege*
2. Décompresser. Puis, lancer Protégé en double cliquant sur run.bat.
3. Il est possible d'étoffer Protégé en installant des plugins. *File => Check for Plugins* => cocher et installer Pellet, SRWL, et SPARQL.

II. Développement d'une ontologie family.owl[1]

1. *Classes et sous-classes*

- File => New.
- Construire l'ontologie *family* selon la figure suivante :

[1] Tirée depuis : http://www-inf.it-sudparis.eu/~gaaloulw/KM/Labs/Lab2/Ontology-based-modeling.htm



2. *Propriétés*

- Datatype Properties : créer les propriétés *name*, *age*, et *nationality* d'une personne. Les propriétés sont *Functional* avec :
  - datatype property *name* with domain **Person** and range xsd:string
  - datatype property *age* with domain **Person** range xsd:int
  - datatype property *nationality* with domain **Person** and range xsd:string

- Object Properties : créer les 11 propriétés suivantes :

  - *isMarriedWith* avec Person comme domain and range
  - *isParentOf* avec **Person** comme domain and range
  - *isFatherOf* qui est sub property of *isParentOf* avec domain **Male** et range **Person**
  - *isMotherOf* qui est sub property of *isParentOf* avec domain **Female** et range **Person**
  - *isSiblingOf* avec domain **Person** et range **Person**
  - *isBrotherOf* qui est sub property of *isSiblingOf* avec domain **Male** et range **Person**
  - *isSisterOf* qui est sub property of *isSiblingOf* aevc domain **Female** et range **Person**
  - *isChildOf* avec domain **Person** et range **Person**
  - *isSonOf* qui est sub property of *isChildOf* avec domain **Male** et range **Person**
  - *isDaughterOf* qui est sub property of *isChildOf* avec domain **Female** et range **Person**

  - *isMarriedWith* et *isSiblingOf* sont symétriques.
  - *isSiblingOf* est transitive.
  - *isChildOf* est l'inverse de *isParentOf*.

3. *Restrictions – Conditions suffisantes et nécessaires*

- Un uncle a la restriction : *is brother of a parent*
- Un grandfather a la restriction : *is father of a parent*
- Une grandmother a la restriction : is mother of a parent
- Un father a la restriction : *isFatherOf* property a au moins une instance
- Une mother a la restriction : *isMotherOf* property a au moins une instance
- Un son a la restriction : *isSonOf* property a au moins une instance
- Une daughter a la restriction : *isDaughterOf* property a au moins une instance
- Un brother a la restriction : *isBrotherOf* property a au moins une instance
- Une sister a la restriction : *isSisterOf* property a au moins une instance

4. *Classes disjointes*

- Male et Female sont disjoints
- Father et Mother sont disjoints
- Son et Daughter sont disjoints
- GandFather et GrandMother sont disjoints

5. *Instances*

- Créer les individus de la classe **Male** :

  - Peter, 70, is*MarriedWith* Marie. French
  - Thomas, 40, *isSonOf* Peter. French
  - Paul, 38 *isSonOf* Peter
  - John, 45, Italian
  - Pedro, 10, *isSonOf* John
  - Tom, 10, *isSonOf* Thomas et Alex
  - Michael, 5, *isSonOf* Thomas et Alex

- Créer les individus de la classe **Female** :

  - Marie, 69, french
  - Sylvie, 30, *isDaughterOf* Marie et Peter
  - Chloé, 18, *isDaughterOf* Marie et Peter
  - Sylvie is*MarriedWith* John
  - Claude, 5, *isDaughterOf* Sylvie, french
  - Alex, 25, is*MarriedWith* Thomas

- Enregistrer votre ontologie OWL sous le nom de family.owl.

III. Vérification et inférence avec Pellet Reasoner

- Ouvrir l'ontologie *humain.owl* dans Protégé (File => Open).

- Lancer le reasoner Pellet en allant sur Barre des menus => *Reasoner* => Cocher Pellet (Incremental) => *Start Reasoner*. Puis, en bas à droite, cocher *Show inferences*.
- Que se passe-t-il ?
- Le reasoner détecte que l'ontologie n'est pas consistante. Une erreur dans les dates de naissance des instances qui devraient être de type xsd:date. *Barre des menus => Reasoner => Explain inconsistent ontology => Explain.*
- Corriger, puis enregistrer.

- Modifier l'ontologie afin que les classes Homme et Femme deviennent disjointes.
- Ajouter à l'instance Marie le type Homme. Enregistrer.
- Que se passe-t-il ? *Barre des menus => Reasoner => Explain inconsistent ontology => Explain.*

IV. SWRL

- Revenir à l'ontologie family.owl et ajouter la classe Adult à l'ontologie
- Ouvrir le SWRL Tab de Protégé. *Window => Tabs => SWRL Tab*.
- Dans SWRL Tab, ajouter une nouvelle règle SWRL en cliquant sur *New*.
- Dans le champ Name, mettre : Def-Adult.
- Ecrire la règle comme suit :

```
Person(?x) ^ age(?x, ?val) ^ swrlb:greaterThan(?val, 18) -> Adult(?x)
```

- Dans le Tab Control, cliquer sur *OWL+SWRL->Drools* puis, sur Run Drools.
- Vérifier dans Inferred Axioms et dans la liste des instances que tous les individus qui ont plus de 18 ans sont devenus aussi des individus de la classe Adult.