Année 2015-2016

Partie « Web sémantique »

Documents autorisés - **durée** indicative : 50 mn – **barême** 10 pts

Exercice 1 : (5 points)

Soit les relations ainsi définies (`#Pizza, #PizzaTopping et #Food` étant des `owl:Class et #hasIngredient` étant une `owl:ObjetProperty`) :

```
<owl:ObjectProperty rdf:about=" #hasIngredient">
    <rdfs:domain rdf:resource="#Food"/>
    <rdfs:range rdf:resource=" #Food"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about=" #hasTopping">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <rdfs:comment xml:lang="en">Note that hasTopping is inverse
functional because isToppingOf is functional</rdfs:comment>
    <rdfs:domain rdf:resource="#Pizza"/>
    <rdfs:range rdf:resource=" #PizzaTopping"/>
    <rdfs:subPropertyOf rdf:resource="#hasIngredient"/>
</owl:ObjectProperty>
```

1. Que signifie que `hasTopping` soit de type `InverseFunctionalProperty` ?
   Que signifie que `isToppingOf` soit de type `FunctionalProperty` ?

Une InverseFunctionalProperty est une propriété qui est l'inverse d'une propriété FunctionalProperty. Une `FunctionalProperty` est une propriété nécessaire, donc "obigatoire", pour toute instance de la classe domaine. `hasToppingOf relie une entité de type #Pizza à une entité de type #PizzaTopping. Comme isToppingOf est inverse de hasTopping, isToppingOf relie une entitié de type #PizzaToping à une entité de type #Pizza.`
`Dire que isToping est functional, c'est dire que pour tout t tel que t rdf:type #PizzaTopping, il doit exiter p tel que p rdf:type #Pizza, et t isToppingOf p.`
`Comme hasTopping est l'inverse de isTopping, pour tout t tel que t rdf:type #PizzaTopping, il doit exiter p tel que p rdf:type #Pizza, et p hasToppingOf t.`

2. Soit le triplet `myPizza #hasTopping myGOrgonzola`
   où `myPizza` et `myGorgonzola` sont deux `owl:Individual`. Sans plus d'information, quels sont les types de ces 2 instances avant de lancer un raisonneur ? après l'avoir lancé ?

expliquer comment un raisonneur exploite des informations liées à rdfs :domain et rdfs :range.

Avant lancement raisonneur : owl :Thing et rdfs :Rsource

Après lancement : myPizza rdf :type #Pizza et myGorgonzola rdf :type #PizzaTopping

Sémantique de rdf :domain et rdf :range

Si R rdf :domain C1 et R rdf :range C2, et si a R b

Alors a rdf :type C1 et b rdf :type C2

Le raisonneur déduit myPizza rdf :type #Pizza du fait que

- myPizza #hasTopping #myGorgonzola et #hasTopping rdfs:domain #Pizza

Le raisonneur déduit myGorgonzola rdf :type #PizzaTopping du fait que

- `myPizza #hasTopping #myGorgonzola` et du fait que `#hasTopping rdfs :range #PizzaTopping`

On définit la classe #CheeseyPizza ci-dessous (sachant que #CheeseTopping est une sous-classe de Food).

```
<owl:Class rdf:about="#CheeseyPizza">
  <rdfs:label xml:lang="pt">PizzaComQueijo</rdfs:label>
  <rdfs:label xml:lang="fr">Pizza au Fromage</rdfs:label>
  <owl:equivalentClass>
     <owl:Class> <owl:intersectionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="#Pizza"/>
          <owl:Restriction>
           <owl:onProperty rdf:resource="#hasTopping"/>
           <owl:someValuesFrom rdf:resource="#CheeseTopping"/>
          </owl:Restriction>
                 </owl:intersectionOf>
     </owl:Class>
  </owl:equivalentClass>
  <rdfs:comment xml:lang="en">Any pizza that has at least 1 cheese
topping.</rdfs:comment>
</owl:Class>
```

3. Reprenons le même triplet myPizza #hasTopping myGorgonzola : quel doit être le type de myGorgonzola si on veut que le raisonneur en déduise que myPizza est de type CheeseyPizza ?

On doit avoir `myGorgonzola rdf :type #CheeseTopping`

4. Quels sont les étiquettes du concept CheeseyPizza?

<rdfs:label xml:lang="pt">PizzaComQueijo</rdfs:label>
<rdfs:label xml:lang="fr">Pizza au Fromage</rdfs:label>

Exercice 2 : (5 points)

Soit l'ontologie suivante écrite avec la notation OWL/Turtle

```
:Entite          rdf:type        owl:Class .
:Fonction        rdfs:subClassOf :Entite .
:Sexe            rdfs:subClassOf :Entite .
:Discipline      rdfs:subClassOf :Entite .
:Cours           rdfs:subClassOf :Entite .
:CoursUniversitaire rdfs:subClassOf :Cours .
:Personne        rdfs:subClassOf :Entite .
:Homme rdfs:subClassOf :Personne .
:Femme rdfs:subClassOf :Personne .
:feminin rdf:type :Genre .
:masculin rdf:type :Genre .
:lea rdf:type :Personne .
:coursOntoN7 rdf:type :CoursUniversitaire .
:coursWebSemantique rdf:type :CoursUniversitaire .
:Genre owl:equivalentClass [
     rdf:type owl:Class ;
     owl:oneOf ( :masculin :feminin )
     ] .
:feminin owl:differentFrom :masculin
:aPourGenre
     rdfs:domain :Personne ;
     rdfs:range :Genre .
:marc :aPourGenre :masculin .
:Femme owl:equivalentClass [
     rdf:type owl:Restriction ;
     owl:onProperty :aPourGenre ;
     owl:hasValue :feminin
     ] .
:Homme owl:equivalentClass [
     rdf:type owl:Restriction ;
     owl:onProperty :aPourGenre ;
     owl:hasValue :masculin
     ] .
```

1. Donner la liste des classes, des propriétés (ObjectProperties) et des instances de cette ontologie.

Classes : :Femme, :Homme, :Genre, :Personne :, :CoursUniversitaire, :Cours, :Entite, :Fonction
Propriétés : :aPourGenre
Instances : :masculin, :feminin, :marc, :lea, :coursOntoN7, :coursWebSemantique

2. Si on lance un raisonneur, quelles classes va-t-il attribuer à l'instance :marc ? pourquoi ?

:Personne car c'est le domaine de la relation :aPourGenre et que :marc :aPourGenre :masculin .

3. Créer une relation :participeCours entre qui indique qu'une personne suit un cours ; par analogie avec la définition de :Homme, créer une classe :Etudiant qui serait définie comme une personne qui suit au moins un cours, représenter que :Marc suit le cours Web sémantique et le cours d'ontologie N7.

```
:participeCours
     rdfs:domain :Personne ;
     rdfs:range :Cours .
:Etudiant rdfs:subClassOf :Personne .
     owl:equivalentClass [
     rdf:type owl:Restriction ;
     owl:onProperty :participeCours ;
     owl:hasValue :masculin
     ] .
:marc :participeCours :coursOntoN7
                      :coursWebSemantique
```

4. Ecrire une requête SPARQL qui retourne tous les étudiants et leurs cours. Cette requête va-t-elle retourner :Marc et les cours qu'il suit ? pourquoi ?

```
SELECT ?Student ?Class
WHERE { ?Student a :Etudiant ;
                    :participeCours ?Class }
```

Non car Marc est de type :Homme mais il n'est pas de type :Etudiant

5. Si on relance le raisonneur, quelles classes va-t-il attribuer à :Marc ? quelles réponses retournera la même requête SPARQL ? qu'en concluez-vous au sujet de l'utilisation de SPARQL ?

```
:Homme et :Etudiant car il est une des 2 entités reliées dans la
relation  :marc :participeCours :coursOntoN7
Et que le domaine de :participeCours est :Etudiant.

Le raisonneur peut en déduire que :marc est un :Etudiant, et la
requête SPARQL aura comme réponse
 :marc :coursOntoN7
 :marc :coursWebSemantique
Le résultat d'une requête SPARQL dépend du fonctionnement du
raisonneur lancé par le serveur où est lancée la requête.
```