Nom :
Prénom :

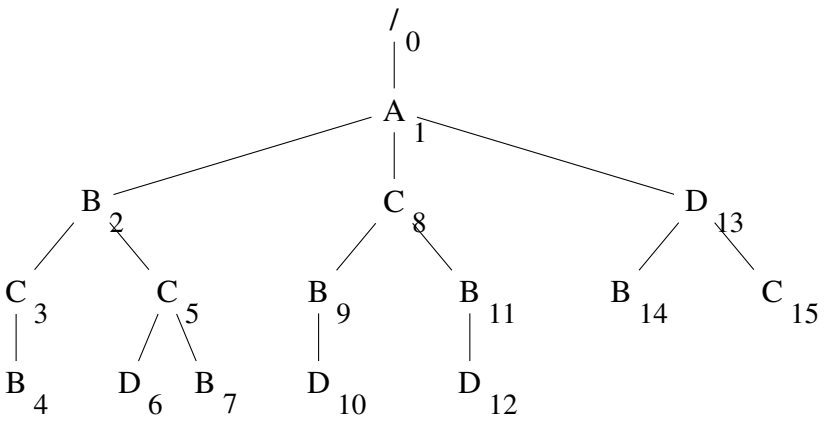
ASWS – Apprentissage Symbolique et Web Sémantique

Examen répartie – 1ère partie du 4 nov 2015
partie Web Sémantique
CORRIGÉ
Documents autorisés

Les téléphones mobiles doivent être éteints et rangés dans les sacs. Le barème sur 20 points (10 questions) n'a qu'une valeur indicative.

1 XML/XPath (3 pts)

Soit donné l'arbre XML suivant où chaque noeud est identifié par un attribut `@id` qui correspond à sa position dans l'ordre du document. L'identifiant de la racine est égal à 0, son élément fils de type `A` a l'identifiant 1 etc.:



Le résultat d'une expression XPath est une liste de noeuds DOM triés dans l'ordre du document et sans doublons. Par exemple, l'expression XPath `/descendant::B/following-sibling::*/@id` retourne les identifiants 8, 11, 13, 15.

Question 1 (1 point)

Donnez pour chaque expression XPath 2.0 ci-dessous la liste des identifiants des noeuds retournés.

Réponse :

```
/descendant::B/following-sibling::*[1]/@id

/descendant::*[not(child::*[2])]/@id

//(* except C)/B/@id

/(descendant::*[not(child::*)] except descendant::*[following::*])
```

Solution:

`/descendant::B/following-sibling::*[1]/@id`: 8, 11, 15
`/descendant::*[not(child::*[2])]/@id`: 3, 4, 6, 7, 9, 10, 11, 12, 14, 15
`//(* except C)/B/@id`: 2, 14
`/(descendant::*[not(child::*)] except descendant::*[following::*])`: 15

Question 2 (2 points)
Est-ce que les expressions XPath suivantes sont équivalentes (indépendamment du document XML interrogé) ?

Réponse :

1. `/descendant::*/parent::C/@id`
   - `/descendant::C[child::*]/@id` oui ☐ non ☐
   - `/descendant::*[child::*[parent::C]]/@id` oui ☐ non ☐
   - `/descendant::C[child::*/child::*]/@id` oui ☐ non ☐
   - `/descendant::C[child::*]/*/../@id` oui ☐ non ☐
2. `/following::*/@id`
   - `/descendant::*/@id` oui ☐ non ☐
   - `/preceding::*/@id` oui ☐ non ☐
   - `/descendant::*/following::*/@id` oui ☐ non ☐
   - `/following::*/preceding::*/following::*/@id` oui ☐ non ☐

Solution:

1. `/descendant::*/parent::C/@id`
   - `/descendant::C[child::*]/@id` oui ■ non ☐
   - `/descendant::*[child::*[parent::C]]/@id` oui ■ non ☐
   - `/descendant::C[child::*/child::*]/@id` oui ☐ non ■
   - `/descendant::C[child::*]/*/../@id` oui ■ non ☐
2. `/following::*/@id`
   - `/descendant::*/@id` oui ☐ non ■
   - `/preceding::*/@id` oui ■ non ☐

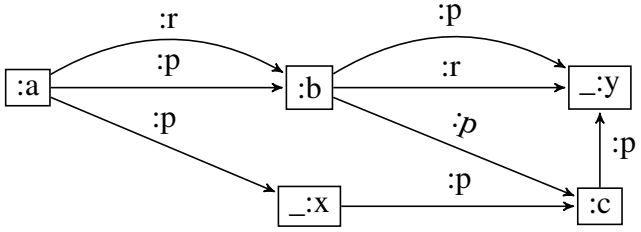
- `/descendant::*/following::*/@id` oui ☐ non ■
- `/following::*/preceding::*/following::*/@id` oui ■ non ☐

L'expression absolue `/following::*/@id` retourne toujours l'ensemble vide (comme `/preceding::*/@i`

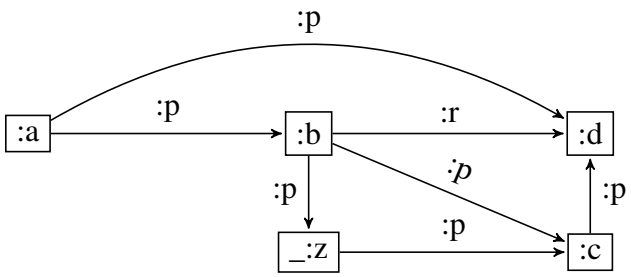
2 Graphes RDF (5 pts)

Soit les deux graphes RDF $G1$ et $G2$ suivants:

G1:



G2:



On suppose que les noeuds et les propriétés appartiennent à l'espace de nom `http://exemple.org/`. Chaque noeud est étiqueté par son identifiant ou par une lettre précédée par '_:' s'il s'agit d'un noeud blanc. Les arcs sont étiquetés par les types des propriétés. *La propriété `:p` est transitive.*

Question 3 (1 point)

Donnez l'expression Turtle qui correspond au graphe $G1$.

Réponse :

Solution:

```
@prefix : <http://exemple.org/> .
:a :r: :b ; :p :b ; :p [ :p :c ] .
:b :p _:y ; :r _:y ; :p :c .
:c :p _:y .
```

ou

```
@prefix : <http://exemple.org/> .
:a :r: :b ; :p :b , [ :p :c ] .
:b :p _:y , :c ; :r _:y .
:c :p _:y .
```

Question 4 (2 points)
Donnez les formes normales $nf(G1)$ et $nf(G2)$ en format Turtle. Attention, la propriété `:p` est transitive.

Réponse :

Expression Turtle $nf(G1)$:

Expression Turtle $nf(G2)$:

Solution:

Expression Turtle $nf(G1)$:

```
@prefix : <http://exemple.org/> .
:a :r :b ; :p :b, :c, _:y .
:b :r _:y; :p     :c, _:y .
:c         :p         _:y .
```

Expression Turtle $nf(G2)$:

```
@prefix : <http://exemple.org/> .
:a :r :b ; :p :b, _:z, :c, :d .
```

```
:b :r :d ; :p     _:z, :c, :d .
_:z        :p          :c, :d .
:c         :p              :d .
```

Question 5 (2 points)
Est-ce que $G1 \models G2$ et/ou $G2 \models G1$? Justifiez votre réponse.

Réponse :

$G1 \models G2$

$G2 \models G1$

Solution:

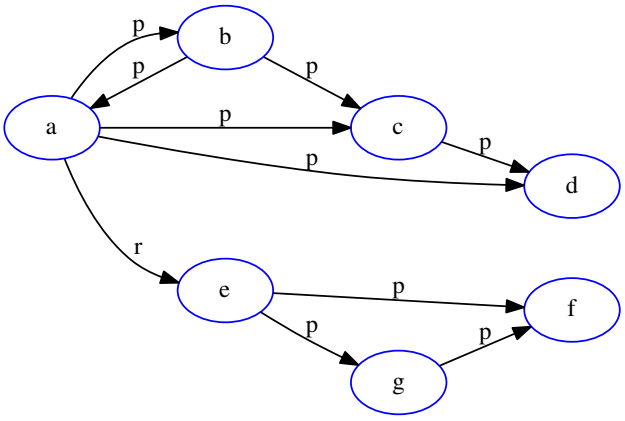
$G1 \models G2$: non; il n'y a pas de mapping m dans $G2$ tel que $(:b : pm(\_ : z)), (m(\_ : z) : p : c)$ dans $G1$

$G2 \models G1$: le triplet $(:a : r : b)$ de $G1$ n'existe pas dans $G2$.

3 SPARQL (5 pts)

Soit le graphe dirigé RDF/S *graphe.ttl*:

```
@prefix : <http://exemple.org/> .
:a :p :b, :c, :d; :r :e .
:b :p :a, :c .
:c :p :d .
:e :p :f, :g .
:g :p :f .
```



Question 6 (5 points)

Donnez les requêtes SPARQL suivantes sur le graphe RDF/S $graphe.ttl$. Vous n'êtes pas obligé de spécifier les espaces de nom utilisés ni le graphe interrogé (expressions SELECT... WHERE ...).

1. Les couples de noeuds connectés par une séquence de propriétés :p de longueur ≥ 2.

Réponse :

Solution:

```
PREFIX : <http://exemple.org/>
SELECT DISTINCT ?x ?y
FROM <graphe2.ttl>
WHERE { ?x :p/:p+ ?y . }
ORDER BY ?x ?y
```

2. Les couples de noeuds connectés par aucun chemin qui contient la propriété `:r`.

Réponse :

Solution:

```
PREFIX : <http://exemple.org/>
SELECT DISTINCT ?x ?y
FROM <graphe2.ttl>
WHERE {{?x :p+ ?y} MINUS { ?x (:p*/:r+/(:p*|:r*)) ?y . }}
```

```
ORDER BY ?x ?y
```

3. Tous les noeuds du graphe.

Réponse :

Solution:

```
PREFIX : <http://exemple.org/>
SELECT DISTINCT ?x
FROM <graphe2.ttl>
WHERE {{ ?x ?p ?y . } UNION { ?y ?p ?x . }}
```

4. Les noeuds sans arc sortant (degré de sortie = 0) en utilisant l'opération MINUS (sans utiliser OPTIONAL/bound()).

Réponse :

Solution:

```
PREFIX : <http://exemple.org/>
SELECT DISTINCT ?y
FROM <graphe2.ttl>
WHERE {{ ?x ?p ?y . } MINUS { ?y ?q ?z . }}
```

4 Règles (2 pts)

Question 7 (2 points)

Définissez un ensemble de *règles forward* qui génèrent un triplet `:a :s :b` entre tous les couples de

noeuds `(:a,:b)` reliés par un chemin qui contient la propriété `:r`.

Réponse :

Solution:

```
@prefix : <http://exemple.org/> .

[trans1: (?x :r ?y) -> (?x :s ?y)]
[trans2: (?x :p ?y)(?y :s ?z) -> (?x :s ?z)]
[trans3: (?x :s ?y)(?y :p ?z) -> (?x :s ?z)]
```

5 Optimisation (5 pts)

On suppose qu'on a chargé un graphe RDF dans une base TDB Jena (tdbloader). Voici un extrait des statistiques de la base:

```
(stats
  (meta
    (timestamp "2015-11-01T19:22:45.874+01:00")
    (run@ "2015/11/01 19:22:45 CET")
    (count 40177))
  ((VAR <rdf:type> <bsbm:Review>) 1000)
  ((VAR <rdf:type> <bsbm:ProductFeature>) 999)
  ((VAR <rdf:type> <bsbm:Product>) 100)
  ((VAR <rdf:type> <bsbm:Producer>) 3)
  (<bsbm:productFeature> 2375)
  (<bsbm:reviewFor> 1000)
  (<bsbm:productProp> 100)
  (<bsbm:producer> 100)
  (<bsbm:country> 54)
  (<bsbm:rating> 717)
  (other 0))
```

Il y a 1000 instances de la classe <bsbm:Review>, 2375 triplets avec la propriété bsbm:productFeature> etc...

On veut exécuter la requête SPARQL suivante:

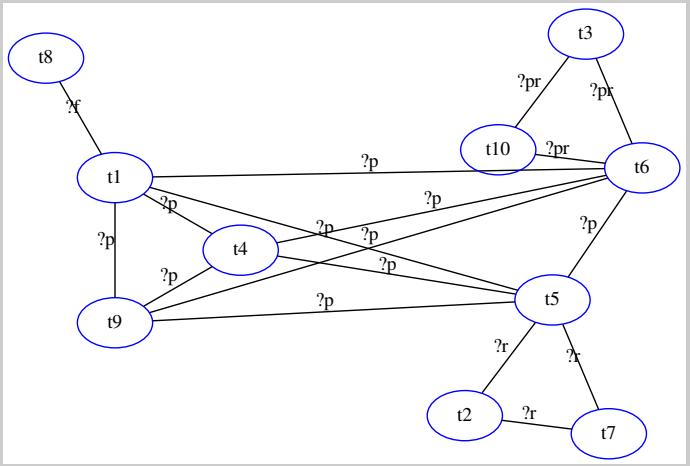
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/bsbm/>

SELECT (COUNT(?r) as ?countreview)
WHERE {
        ?p bsbm:productFeature ?f .
        ?r bsbm:rating ?ra .
        ?pr bsbm:country ?c .
        ?p bsbm:productProp ?pn .
        ?r bsbm:reviewFor ?p .
        ?p bsbm:producer ?pr .
        ?r rdf:type bsbm:Review .
        ?f rdf:type bsbm:ProductFeature .
        ?p rdf:type bsbm:Product .
        ?pr rdf:type bsbm:Producer .
}
```

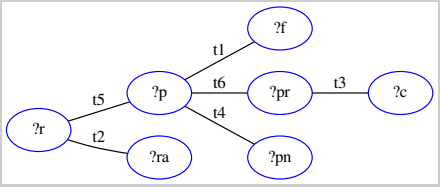
Question 8 (2 points)

Dessinez le graphe de jointure et le graphe de variables de la requête précédente. Vous pouvez numéroter les triplets dans l'ordre de leur apparition dans la requête ($t1$, $t2$, ..., $t10$).

Réponse :

Solution:





Question 9 (1 point)

Expliquez pourquoi l'évaluation de la requête prend beaucoup de temps si on n'applique les filtres dans l'ordre d'apparition dans la clause WHERE (pas d'optimisation, option `none.opt`).

Réponse :

Solution:

Voici l'ordre d'évaluation de jointures qu'on obtient en utilisant la stratégie **fixed.opt** (heuristiques sans statistiques):

```
(?r rdf:type <bsbm:Review>)
(?r <bsbm:rating> ?rating)
(?r <bsbm:reviewFor> ?p)
(?p rdf:type <bsbm:Product>)
(?p <bsbm:productProp> ?pn)
(?p <bsbm:productFeature> ?f)
(?f rdf:type <bsbm:ProductFeature>)
(?p <bsbm:producer> ?pr)
(?pr rdf:type <bsbm:Producer>)
(?pr <bsbm:country> ?c)
```

Question 10 (2 points)

Est-ce que ce plan est optimal ? Expliquez comment il pourrait être amélioré en utilisant les statistiques (**stats.opt**).

Réponse :

Solution:

```
(?pr rdf:type <bsbm:Producer>)
(?pr <bsbm:country> ?c)
(?p <bsbm:producer> ?pr)
(?p rdf:type <bsbm:Product>)
(?p <bsbm:productProp> ?pn)
(?p <bsbm:productFeature> ?f)
(?f rdf:type <bsbm:ProductFeature>)
(?r <bsbm:reviewFor> ?p)
(?r rdf:type <bsbm:Review>)
(?r <bsbm:rating> ?rating)
```