M1IF04 GDW - TD3
Correction
RDFS et inférence

Exercice 1:

On considère le graphe suivant (syntaxe n3):

```
@prefix ucbl: <http://univ-lyon1.fr#> .
@prefix ue:   <http://univ-lyon1.fr/ue#> .
@prefix etu:  <http://univ-lyon1.fr/etudiant#> .
@prefix form: <http://univ-lyon1.fr/formation#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ue:mif18 ucbl:formation form:m1if.
ue:mif17 ucbl:formation form:m1if;
         ucbl:departement ucbl:informatique .
ue:mif16 ucbl:formation form:m1if;
         ucbl:departement ucbl:informatique .
ue:mif13 ucbl:formation form:m1if .
etu:1234567 ucbl:inscrit ue:mif18;
            ucbl:nom "Alice" .
etu:2345678 ucbl:inscrit ue:mif17 , ue:mif13 ;
            ucbl:binome etu:3456789;
            ucbl:nom "Basile" .
etu:3456789 ucbl:inscrit ue:mif17 ;
            ucbl:nom "Charlotte" .
etu:4567890 ucbl:inscrit ue:mif13 , ue:mif16 ;
            ucbl:binome etu:1234567 ;
            ucbl:nom "Damien" .
form:m1if ucbl:departement ucbl:informatique .
form:m2ti ucbl:departement ucbl:informatique .
ucbl:inscrit rdfs:domain ucbl:etudiant .
ucbl:inscrit rdfs:range ucbl:ue .
```

et sa représentation graphique dans la figure 1
On considère les règles RDFS suivantes (nommées entre parenthèses):

$$\frac{P \texttt{ rdfs:domain } T \quad \text{et} \quad S\ P\ O}{S \texttt{ rdf:type } T} \quad (RDFSDomain)$$

$$\frac{P \texttt{ rdfs:range } T \quad \text{et} \quad S\ P\ O}{O \texttt{ rdf:type } T} \quad (RDFSRange)$$

ainsi que les règles métier suivantes:

$$\frac{E \texttt{ ucbl:binome } E2}{E2 \texttt{ ucbl:binome } E} \quad (SymBinome)$$

$$\frac{E \texttt{ ucbl:binome } E2 \quad \text{et} \quad E \texttt{ ucbl:inscrit } UE}{E2 \texttt{ ucbl:inscrit } UE} \quad (BinomeInscrit)$$

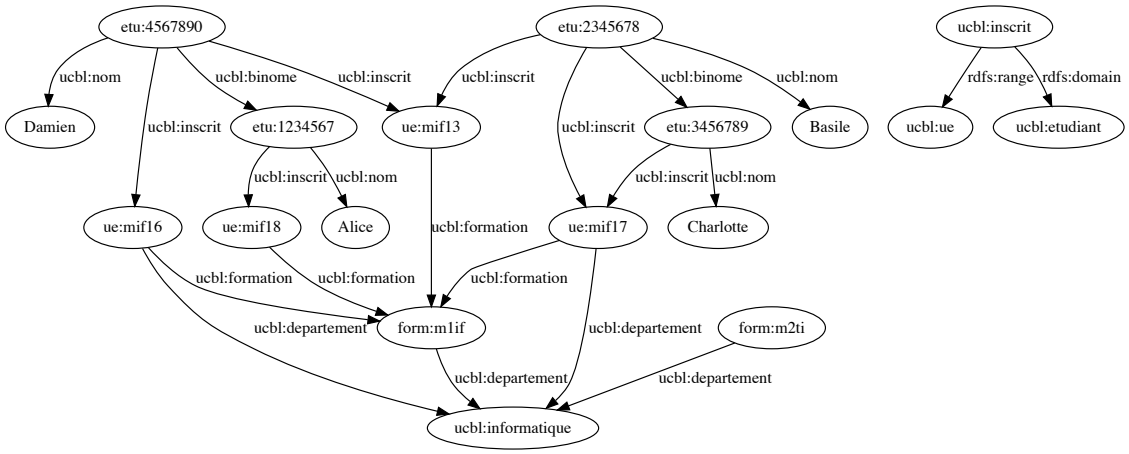

Figure 1: Représentation du graphe

1. Donner une/des règle(s) métier permettant de déduire qu'une UE dans une formation doit être dans le même département que cette formation.

   Correction:

   $$\frac{UE \texttt{ ucbl:formation } F \quad \text{et} \quad F \texttt{ ucbl:departement } D}{UE \texttt{ ucbl:departement } D} \quad (DepartementUE)$$

2. Saturer le graphe en utilisant les règles pour déduire tous les nouveaux triplets possibles.

   Correction:
   Par ($DepartementUE$), on déduit:
   ```
   ue:mif13 ucbl:departement ucbl:informatique.
   ue:mif18 ucbl:departement ucbl:informatique.
   ```
   Par ($SymBinome$), on déduit:
   ```
   etu:1234567 ucbl:binome etu:4567890.
   etu:3456789 ucbl:binome etu:2345678.
   ```
   Par ($BinomeInscrit$), on déduit:
   ```
   etu:1234567 ucbl:inscrit ue:mif16.
   etu:1234567 ucbl:inscrit ue:mif13.
   etu:4567890 ucbl:inscrit ue:mif18.
   etu:3456789 ucbl:inscrit ue:mif13.
   ```
   Par ($RDFSDomain$), on déduit:
   ```
   etu:1234567 rdf:type ucbl:etudiant.
   etu:2345678 rdf:type ucbl:etudiant.
   etu:3456789 rdf:type ucbl:etudiant.
   etu:4567890 rdf:type ucbl:etudiant.
   ```
   Par ($RDFSRange$), on déduit:
   ```
   ue:mif13 rdf:type ucbl:ue.
   ue:mif16 rdf:type ucbl:ue.
   ue:mif17 rdf:type ucbl:ue.
   ue:mif18 rdf:type ucbl:ue.
   ```

3. On considère la requête suivante:

```
1 PREFIX ucbl: <http://univ-lyon1.fr#> .
2 PREFIX ue:   <http://univ-lyon1.fr/ue#> .
```

```
3 PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4
5 SELECT ?e WHERE {
6   ?e ucbl:inscrit ue:mif18 .
7   ?e rdf:type ucbl:etudiant .
8 }
```

(a) Répondre à la requête sur le graphe saturé précédent.

Correction:

?e
etu:1234567
etu:4567890

(b) Répondre à la requête sur le graphe de départ, mais en utilisant le chaînage arrière pour déduire à la volée les triplets intéressants via le chaînage arrière.

Correction: (✓↔ trouvé dans le graphe)

`?e ucbl:inscrit ue:mif18.  ?e rdf:type ucbl:etudiant`

← `etu:1234567 ucbl:inscrit ue:mif18.`✓ (`?e` ⇐ `etu:1234567`)
`etu:1234567 rdf:type ucbl:etudiant.`

← `etu:1234567 ?x1 ?x2.` (*RDFSDomain*)
`?x1 rdfs:domain ucbl:etudiant.`

← `etu:1234567 ucbl:inscrit ue:mif18.`✓ (`?x1` ⇐ `ucbl:inscrit`)
`ucbl:inscrit rdfs:domain ucbl:etudiant.`✓ (`?x2` ⇐ `ue:mif18`)
Résultat trouvé

← `?x3 ucbl:binome ?e.` (*BinomeInscrit*)
`?x3 ucbl:inscrit ue:mif18.`
`?e rdf:type ucbl:etudiant.`

← `etu:1234567 ucbl:binome ?e.`
`etu:1234567 ucbl:inscrit ue:mif18.`✓ (`?x3` ⇐ `etu:1234567`)
`?e rdf:type ucbl:etudiant.`

← `?e ucbl:binome etu:1234567.` (*SymBinome*)
`?e rdf:type ucbl:etudiant.`

← `etu:4567890 ucbl:binome etu:1234567.`✓ (`?e` ⇐ `etu:4567890`)
`etu:4567890 rdf:type ucbl:etudiant.`

← `etu:4567890 ?x4 ?x5.` (*RDFSDomain*)
`?x4 rdfs:domain ucbl:etudiant.`

← `etu:4567890 ucbl:inscrit ue:mif16.`✓ (`?x4` ⇐ `ucbl:inscrit`)
`ucbl:inscrit rdfs:domain ucbl:etudiant.`✓ (`?x5` ⇐ `ue:mif16`)
Résultat trouvé

(c) Expliquer pourquoi, quand on interroge ce graphe, la ligne 7 est inutile dans cette requête.

Correction: Le graphe contient le triplet `ucbl:inscrit rdfs:domain ucbl:etudiant`. Comme on a nécessairement dans la réponse un triplet S `ucbl:inscrit` O à cause de la ligne 6, on sait que le triplet `?e rdf:type ucbl:etudiant` est systématiquement déductible pour toutes les réponses.

(d) Réécrire la requête de façon à y répondre sur le graphe d'origine mais en prenant les inférences en compte.

Correction: Si on supprime le type:

```
PREFIX ucbl:  <http://univ-lyon1.fr#> .
PREFIX ue:  <http://univ-lyon1.fr/ue#> .
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

SELECT ?e WHERE {
  { ?e ucbl:inscrit ue:mif18 .  }
  UNION
  {
    { { ?e2 ucbl:binome ?e.  } UNION { ?e ucbl:binome ?e2.  } }
    ?e2 ucbl:inscrit ue:mif18.
  }
}
```

Attention cependant, cette réécriture suffit pour ces données, mais pas dans le cas général. Si on considère des chaînes de binômes , on risque de rater des résultats. Par exemple A en binôme avec B et B en binôme avec C, fait que si A est inscrit dans une UE, B et donc C doit l'être aussi.