Introduction à RDF

Michel Gagnon

Plan

- Présentation de RDF
- Sérialisations N-Triples, RDF/XML et Turtle
- Sémantique et inférence

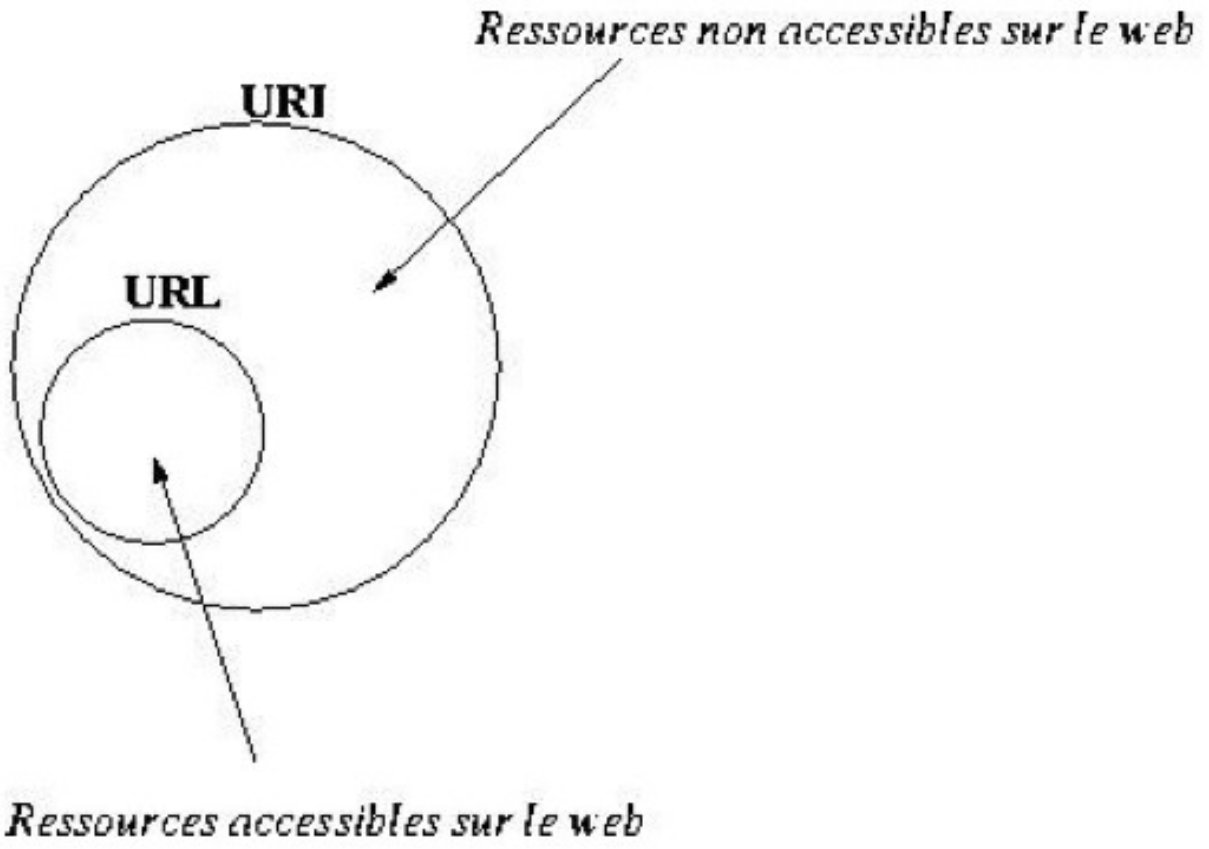
RDF

- Modèle de données pour décrire des ressources du web
- Graphe:
  - les noeuds représentent des ressources
  - les arcs représentent des relations entre ces ressources
- Les ressources sont représentées par leur URI

RDF

- Le graphe est représenté par un ensemble d'énoncés (statements)
- Un énoncé est un triplet <S, P, O>, où
  - S est le sujet
  - P est le prédicat (une propriété)
  - O est l'objet (la valeur de la propriété pour le sujet en question)

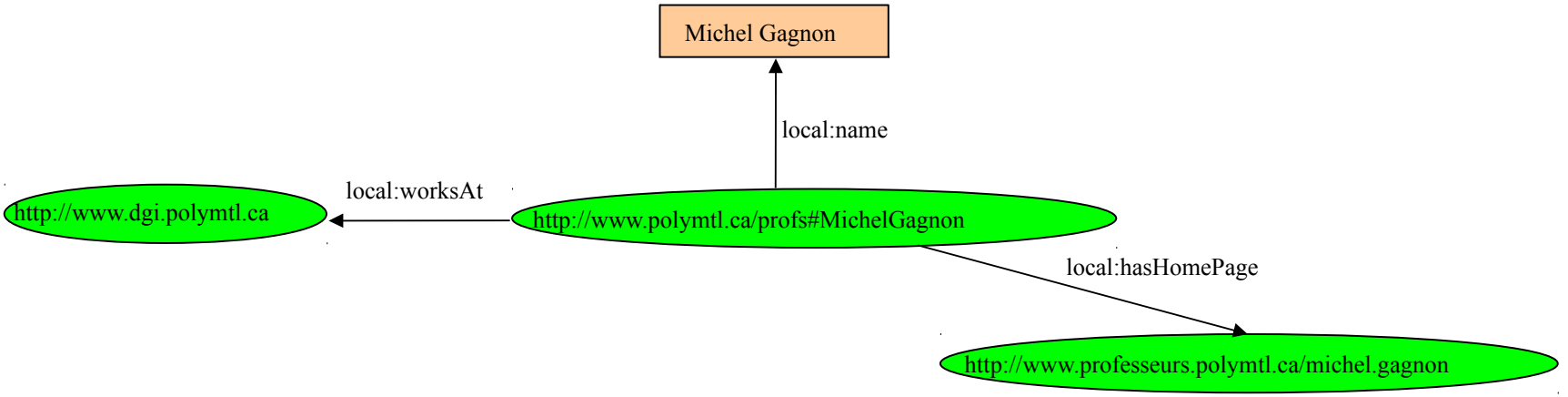
URI


Ressources non accessibles sur le web
URI
URL
Ressources accessibles sur le web

URIref

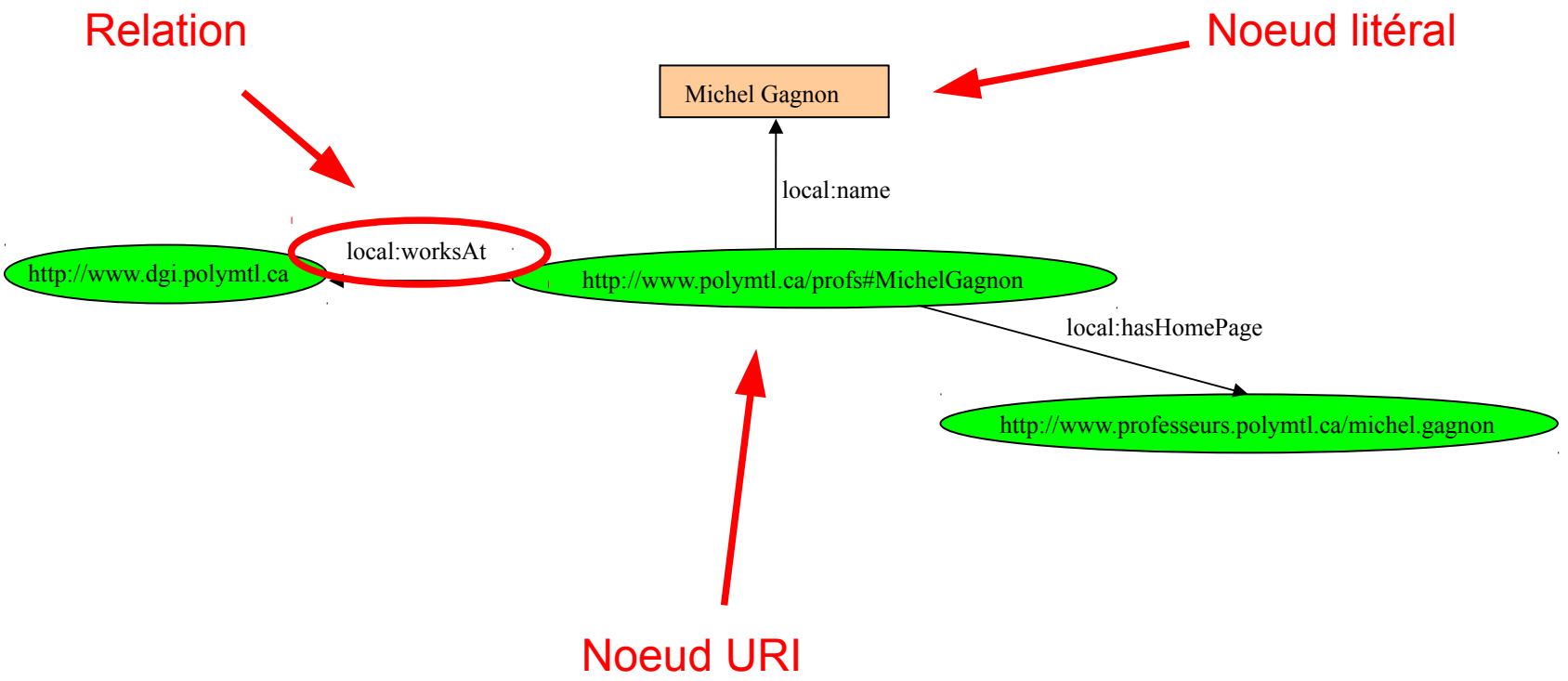
- Plus précisément, les ressources sont identifiées par des URIrefs, c'est-à-dire URI + identificateur de fragment:
  - http://www.polymtl.ca/Profs (URI)
  - #MichelGagnon (Fragment)
  - http//www.polymtl.ca/Profs#MichelGagnon
- En HTML, ceci permet de désigner une section dans un document, alors que pour RDF il ne s'agit que d'un nom donné à une ressource

RDF – Exemple

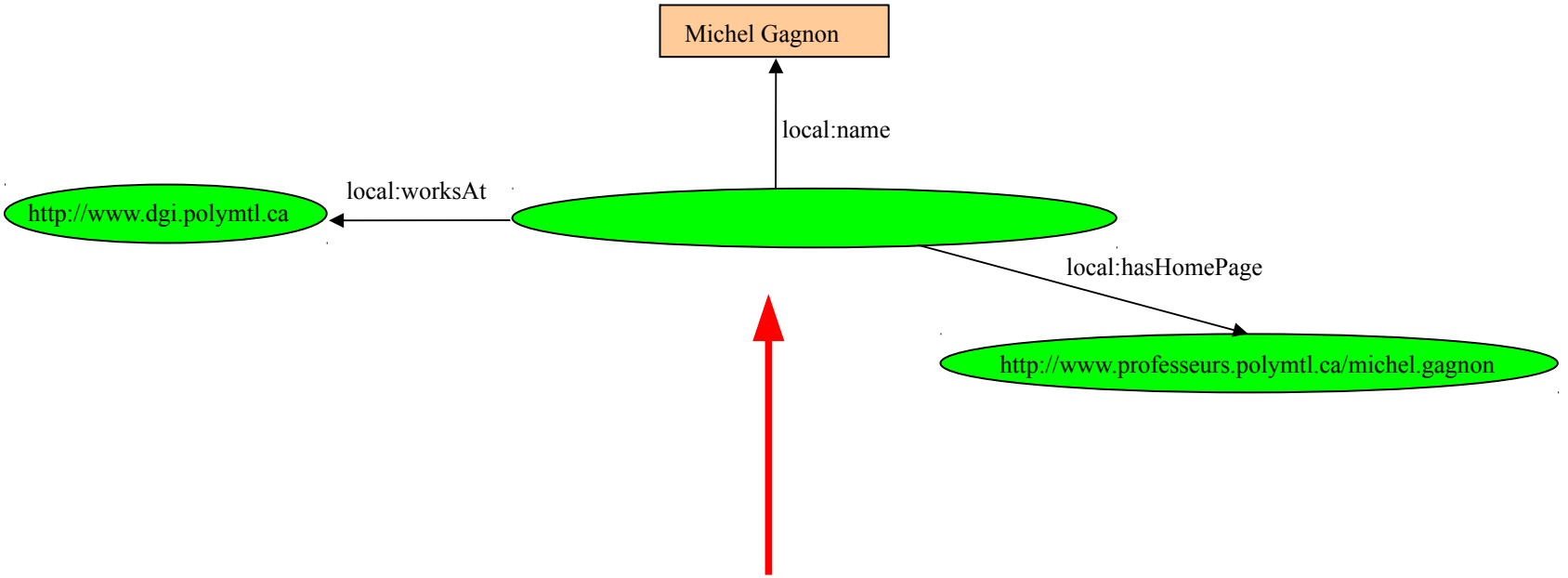


local: http://www.polymtl.ca/vocab#

RDF – Exemple


Relation
Noeud litéral
Michel Gagnon
local:name
local:worksAt
http://www.dgi.polymtl.ca
http://www.polymtl.ca/profs#MichelGagnon
local:hasHomePage
http://www.professeurs.polymtl.ca/michel.gagnon
Noeud URI

RDF – Exemple



Un nœud peut être vide

Syntaxe abstraite

- Collection de triplets
- Une telle collection forme un *graphe RDF*
- Puisque les propriétés sont désignées par des URI, on peut donc les décrire comme n'importe quelle ressource
- Un noeud peut être :
  - Une URI
  - Un littéral (typé ou non)
  - Un nœud vide (il désigne en quelque sorte une ressource dont on ne connait pas le nom)
- Deux types de littéraux :
  - Simple: “Michel Gagnon”
  - Typé: “10”^^xsd:integer

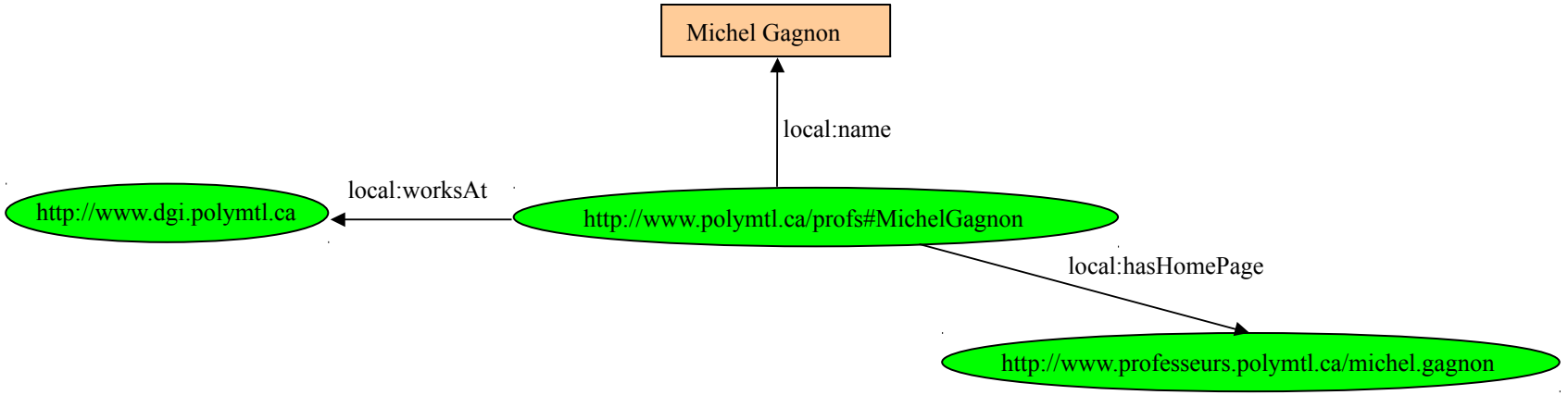
Sérialisation N-Triples

- Un graphe RDF est représenté par une collection de triplets de la forme

```
sujet prédicat objet .
```

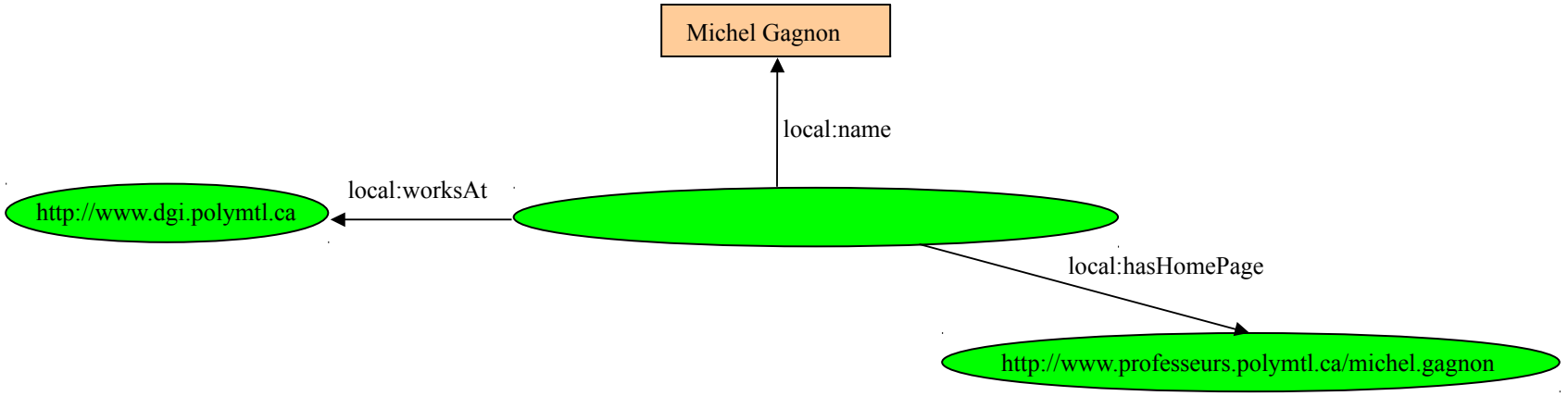
- Si un élément est une URI, on le met entre crochets : <>
- S'il s'agit d'un noeud vide, on utilise la forme `_:nom` où `nom` est un identificateur unique pour ce noeud vide
- Un littéral est représenté tel quel

Sérialisation N - Triples



<http://www.polymtl.ca/profs#MichelGagnon> <http://www.polymtl.ca/vocab#hasHomePage> <http://www.professeurs.polymtl.ca/michel.gagnon> .
<http://www.polymtl.ca/profs#MichelGagnon> <http://www.polymtl.ca/vocab#worksAt > <http://www.dgi.polymtl.ca> .
<http://www.polymtl.ca/profs#MichelGagnon> <http://www.polymtl.ca/vocab#name> "Michel Gagnon" .

Sérialisation N - Triples



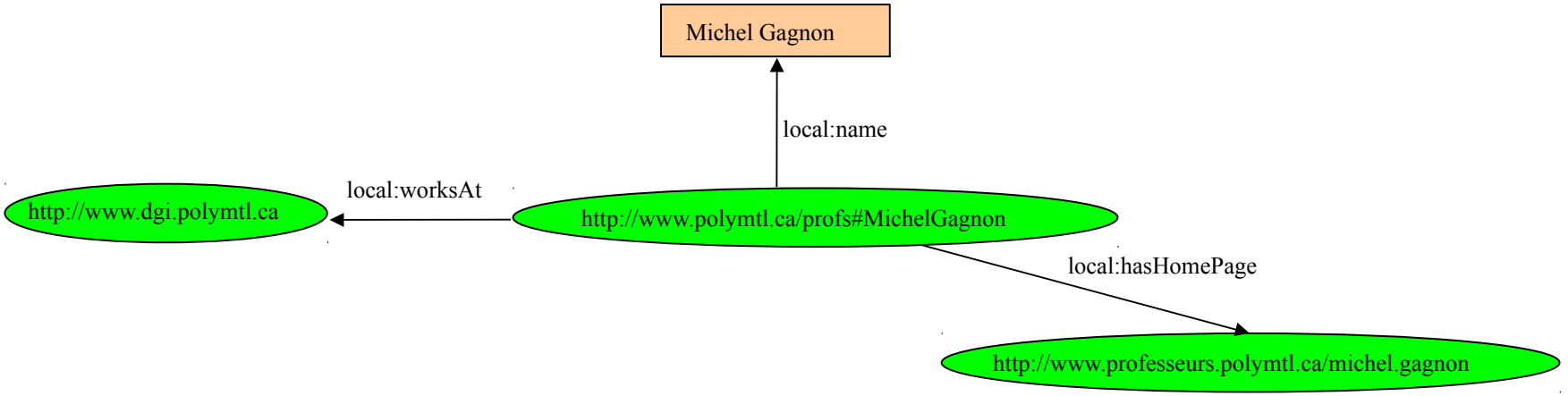

```
_:p43  <http://www.polymtl.ca/vocab#hasHomePage> <http://www.professeurs.polymtl.ca/michel.gagnon> .
_:p43  <http://www.polymtl.ca/vocab#worksAt > <http://www.dgi.polymtl.ca> .
_:p43  <http://www.polymtl.ca/vocab#name>  "Michel Gagnon"  .
```

Sérialisation RDF/XML

- Utilise les espaces de nommage
- Balise `rdf:Description` pour regrouper les descriptions d'une ressource
- Pour un nœud vide, on retire l'attribut *about*
- Pour étiqueter un noeud vide, on utilise la balise

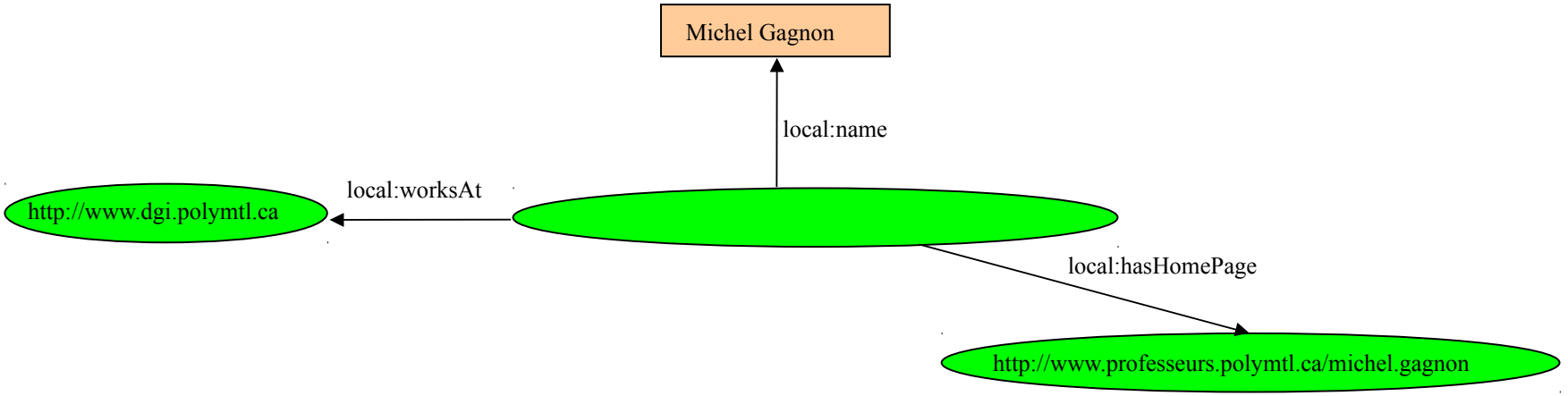
  `rdf:nodeID`
- Pour représenter un littéral typé, on utilise l'attribut `rdf:datatype` dans le prédicat qui relie la ressource à ce littéral
- Il y a souvent plusieurs manières de représenter le même graphe RDF

Sérialisation RDF/XML



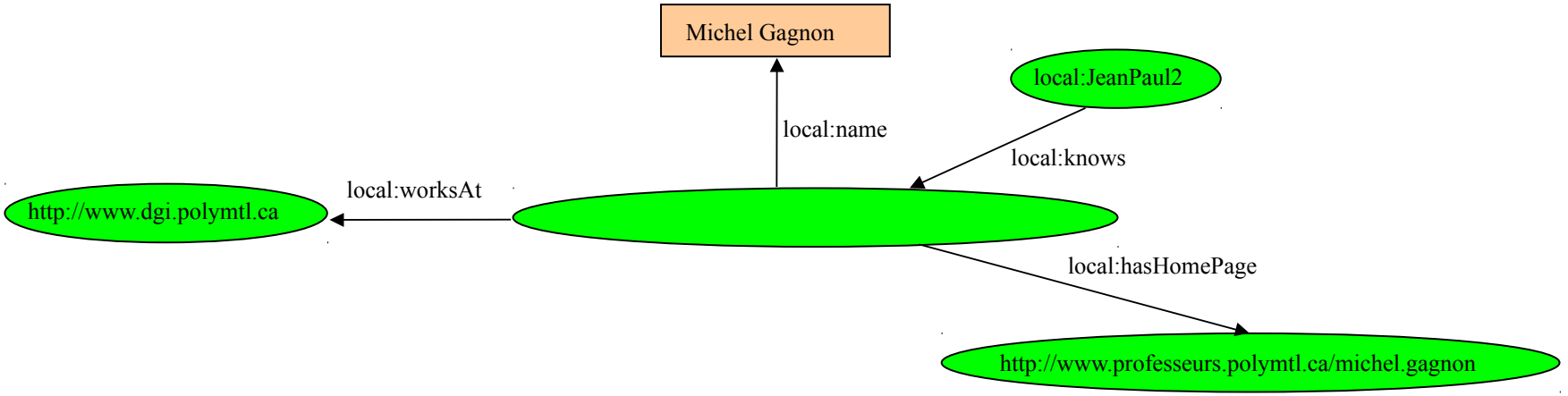

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:local="http://www.polymtl.ca/vocab#">
   <rdf:Description rdf:about="http://www.polymtl.ca/profs#MichelGagnon">
      <local:hasHomePage
            resource="http://www.professeurs.polymtl.ca/michel.gagnon"/>
      <local:worksAt resource="http://www.dgi.polymtl.ca"/>
      <local:name> Michel Gagnon  </local:name>
   </rdf:Description>
</rdf:RDF>
```

Sérialisation RDF/XML



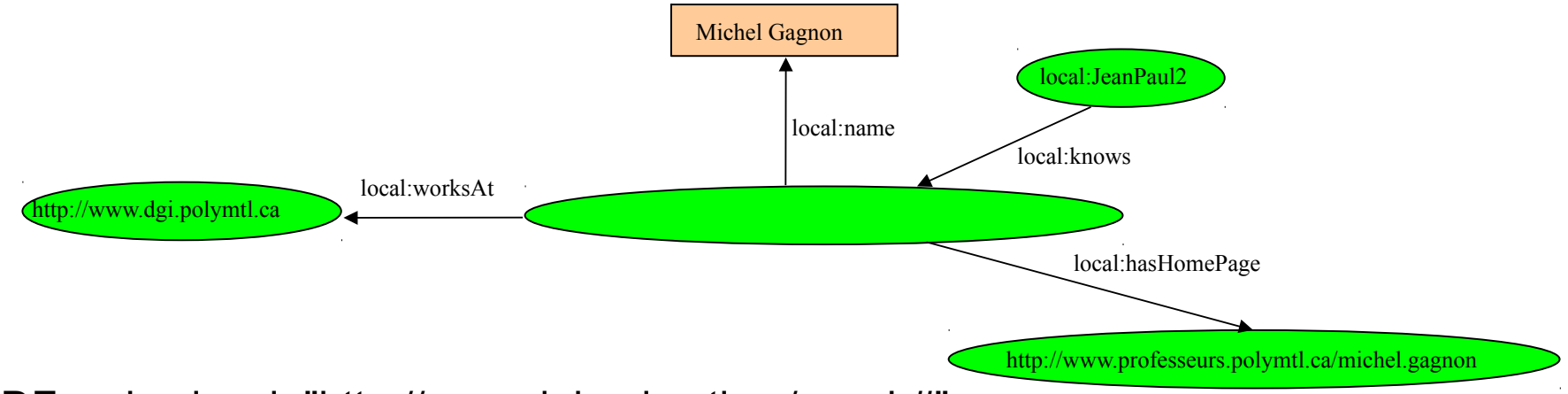

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:local="http://www.polymtl.ca/vocab#">
   <rdf:Description>
      <local:hasHomePage
            resource="http://www.professeurs.polymtl.ca/michel.gagnon"/>
      <local:worksAt resource="http://www.dgi.polymtl.ca"/>
      <local:name> Michel Gagnon  </local:name>
   </rdf:Description>
</rdf:RDF>
```

Sérialisation RDF/XML



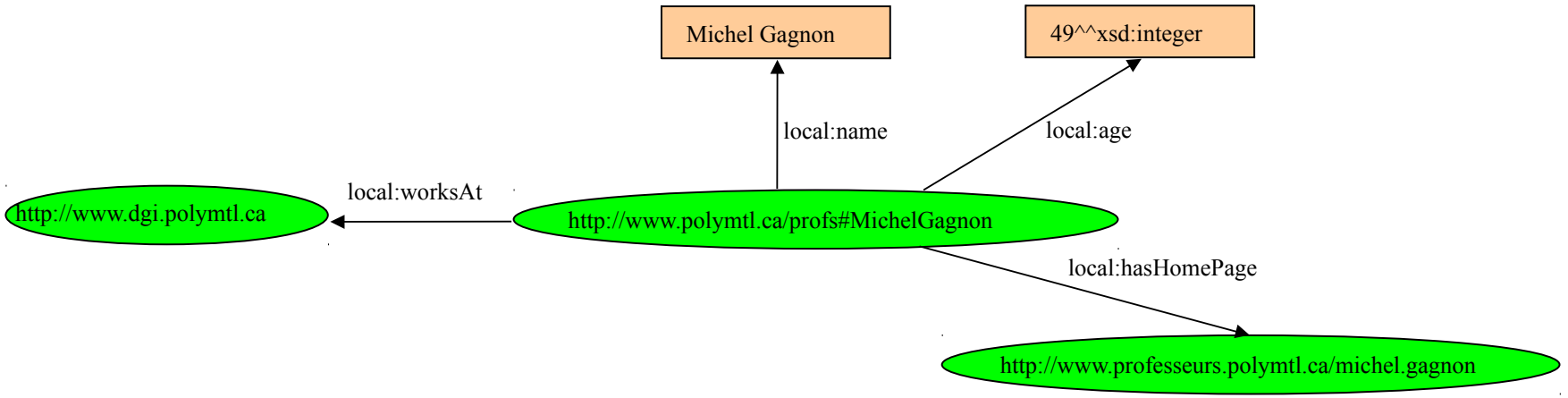

```
<rdf:RDF xmlns:local="http://www.dgi.polymtl.ca/vocab#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:nodeID="n2">
    <local:hasHomePage rdf:resource="http://www.professeurs.polymtl.ca/michel.gagnon" />
    <local:worksAt rdf:resource="http://www.dgi.polymtl.ca" />
    <local:name>Michel Gagnon</local:name>
</rdf:Description>
<rdf:Description rdf:about="http://www.dgi.polymtl.ca/vocab#JeanPaul2">
    <local:connait rdf:nodeID="n2" />
</rdf:Description>
</rdf:RDF>
```

Sérialisation RDF/XML




```
<rdf:RDF xmlns:local="http://www.dgi.polymtl.ca/vocab#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://www.dgi.polymtl.ca/vocab#JeanPaul2">
  <local:knows>
    <rdf:Description>
      <local:hasHomePage rdf:resource="http://www.professeurs.polymtl.ca/michel.gagnon" />
      <local:worksAt rdf:resource="http://www.dgi.polymtl.ca" />
      <local:name>Michel Gagnon</local:name>
    </rdf:Description>
  </local:knows>
</rdf:Description>
</rdf:RDF>
```

Sérialisation RDF/XML

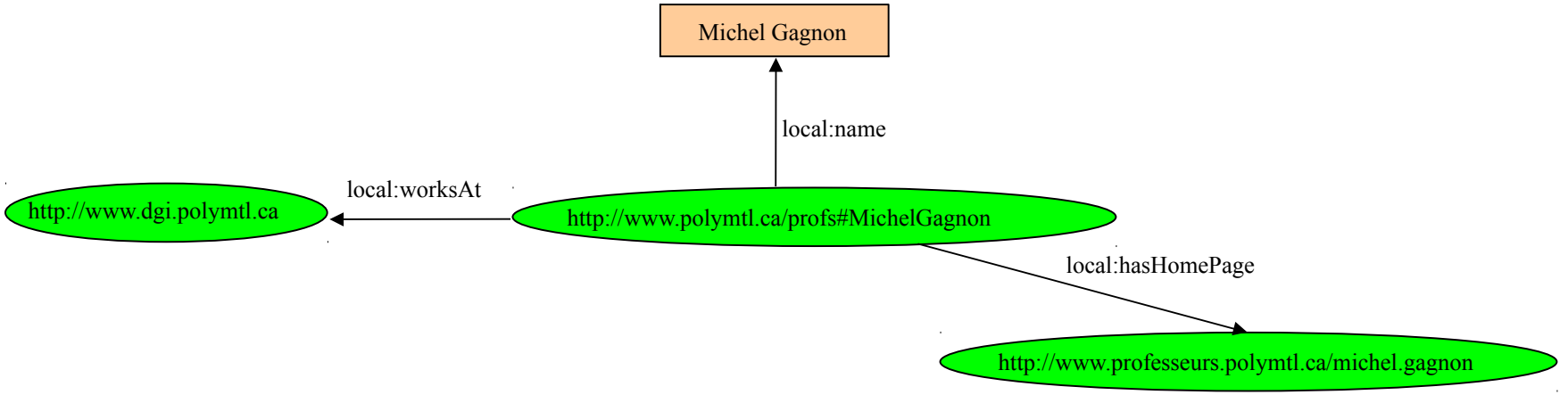



```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:local="http://www.polymtl.ca/vocab#">
   <rdf:Description rdf:about="http://www.polymtl.ca/profs#MichelGagnon">
      <local:hasHomePage
            resource="http://www.professeurs.polymtl.ca/michel.gagnon"/>
      <local:worksAt resource="http://www.dgi.polymtl.ca"/>
      <local:age rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">49</local:age>
      <local:name> Michel Gagnon  </local:name>
   </rdf:Description>
</rdf:RDF>
```

Sérialisation Turtle

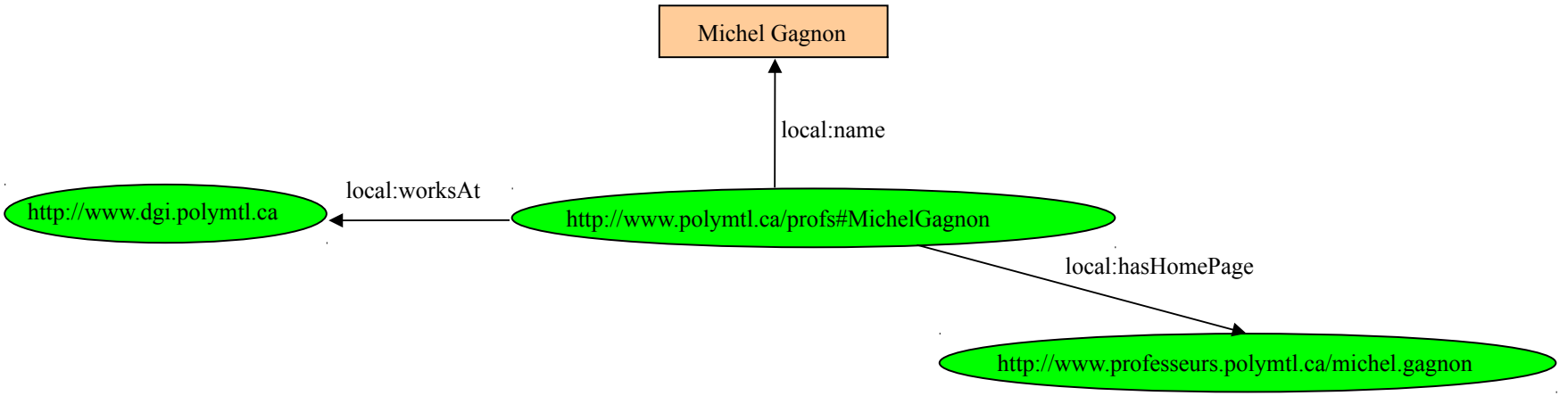
- Permet de spécifier des préfixes
- Permet de combiner des descriptions d'une même ressource :
  - On utilise ; pour grouper des triplets concernant un même sujet
  - On utilise , pour grouper plusieurs instances d'une propriété concernant un même sujet
- Noeud vide représenté par les crochets [ ]
- Toutes les descriptions relatives à un noeud vide peuvent être placées à l'intérieur des crochets

Sérialisation Turtle



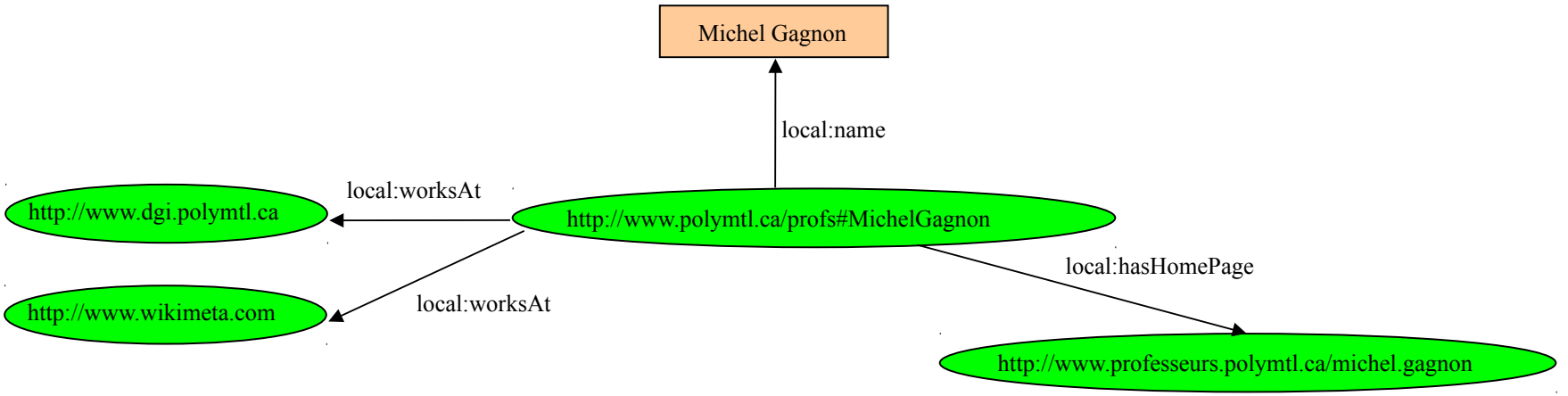

```
@prefix  local: <http://www.polymtl.ca/vocab#> .
@prefix  prof:  <http://www.polymtl.ca/profs#> .

prof:MichelGagnon
    local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> .

prof:MichelGagnon local:worksAt  <http://www.dgi.polymtl.ca> .

prof:MichelGagnon local:name "Michel Gagnon" .
```

Sérialisation Turtle



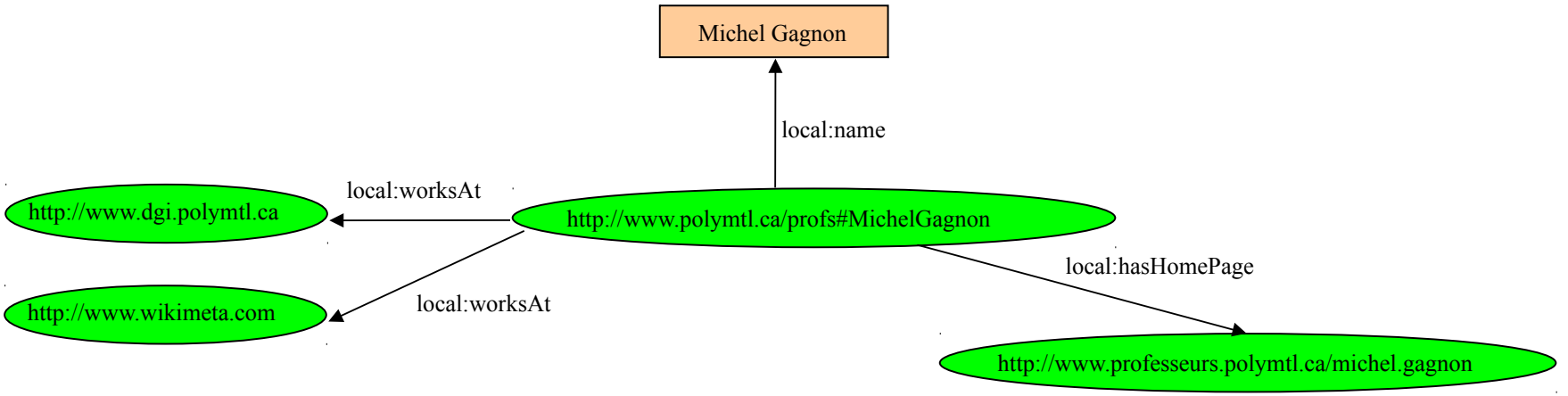

```
@prefix  local: <http://www.dgi.polymtl.ca/vocab#> .
@prefix  prof:  <http://www.dgi.polymtl.ca/profs#> .

prof:MichelGagnon
    local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> ;
    local:worksAt  <http://www.dgi.polymtl.ca> ;
    local:name "Michel Gagnon" .
```

Sérialisation Turtle



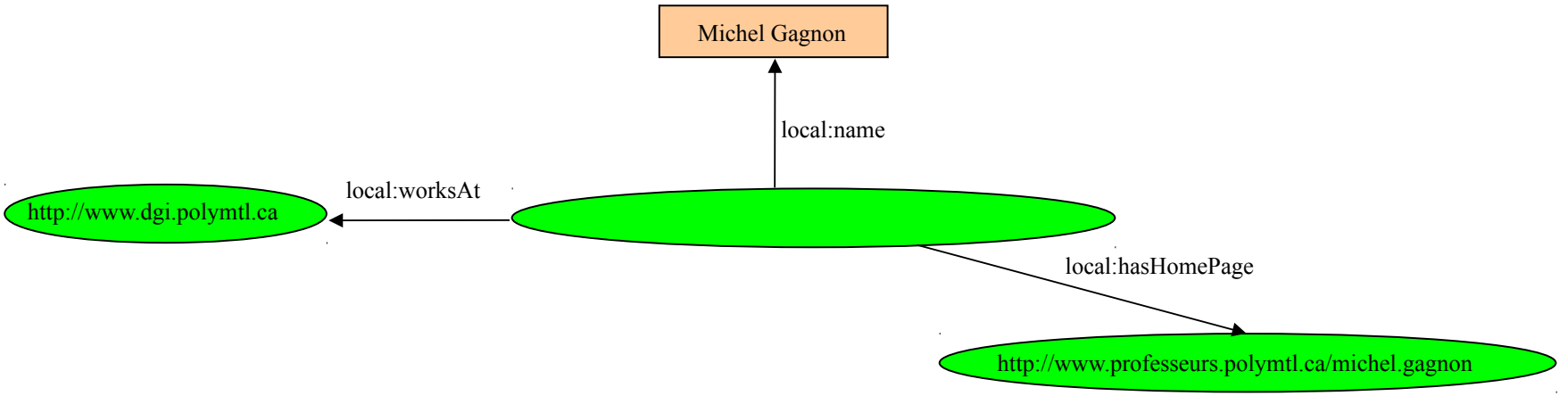

```
@prefix  local: <http://www.dgi.polymtl.ca/vocab#> .
@prefix  prof:  <http://www.dgi.polymtl.ca/profs#> .

prof:MichelGagnon
    local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> ;
    local:worksAt  <http://www.dgi.polymtl.ca> ;
    local:worksAt  <http://www.wikimeta.com> ;
    local:name "Michel Gagnon" .
```

Sérialisation Turtle



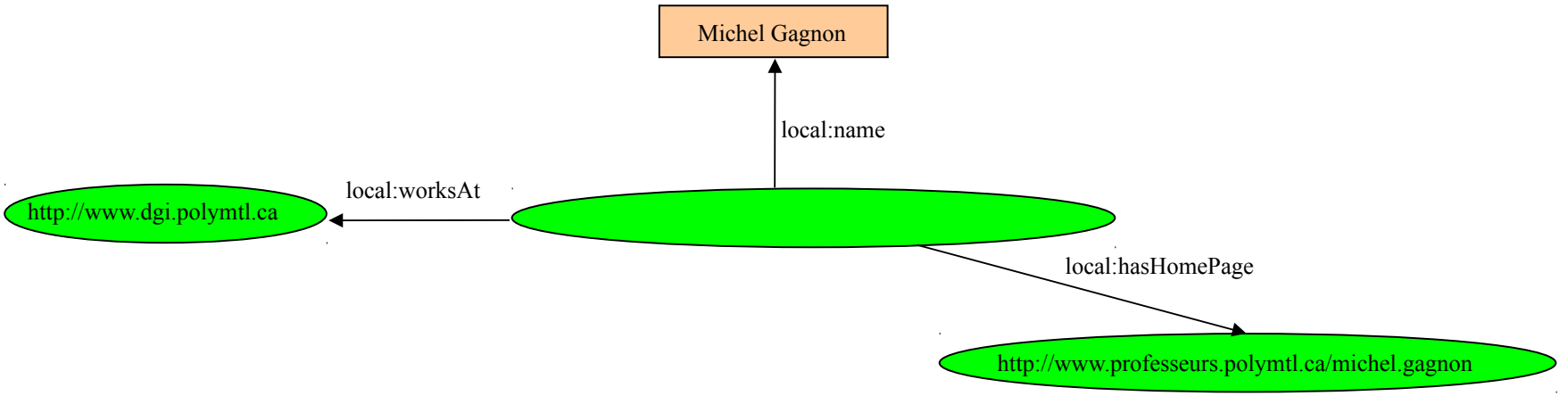

```
@prefix  local: <http://www.dgi.polymtl.ca/vocab#> .
@prefix  prof:  <http://www.dgi.polymtl.ca/profs#> .

prof:MichelGagnon
    local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> ;
    local:worksAt  <http://www.dgi.polymtl.ca> ,
                   <http://www.wikimeta.com> ;
    local:name "Michel Gagnon" .
```

Sérialisation Turtle



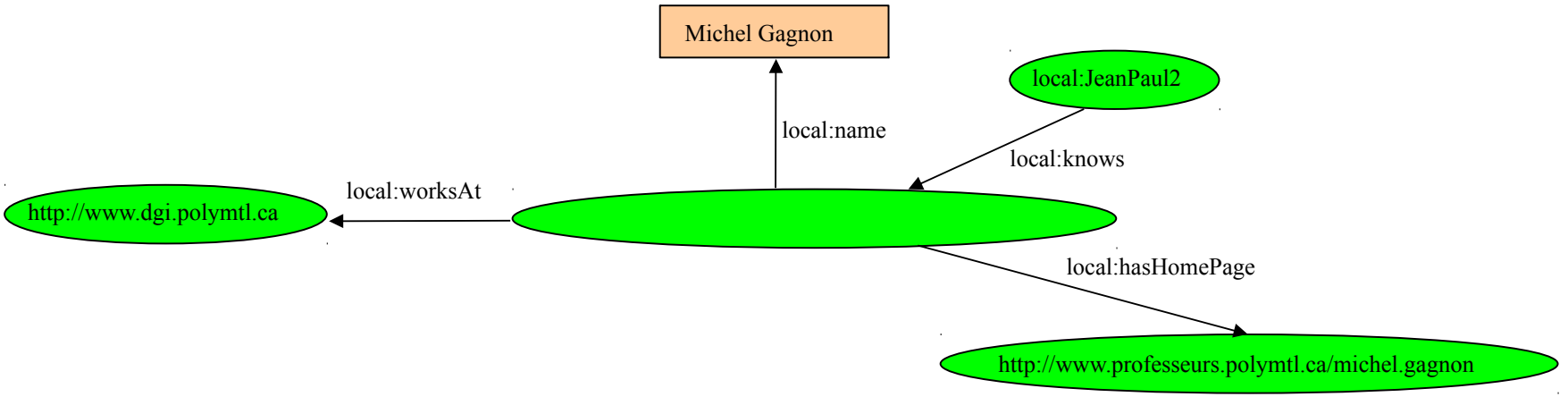

```
@prefix  local: <http://www.dgi.polymtl.ca/vocab#> .
@prefix  prof:  <http://www.dgi.polymtl.ca/profs#> .

[ ]
    local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> ;
    local:worksAt  <http://www.dgi.polymtl.ca> ;
    local:name "Michel Gagnon" .
```

Sérialisation Turtle



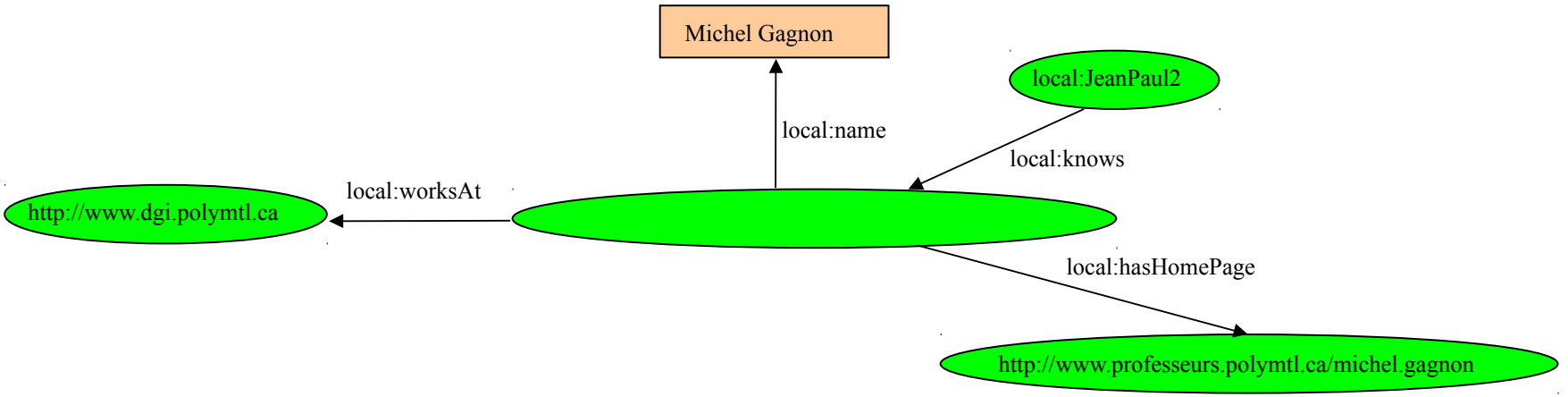

```
@prefix  local: <http://www.dgi.polymtl.ca/vocab#> .
@prefix  prof:  <http://www.dgi.polymtl.ca/profs#> .

[
    local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> ;
    local:worksAt  <http://www.dgi.polymtl.ca> ;
    local:name "Michel Gagnon" .
] .
```

Sérialisation Turtle




```
@prefix  local: <http://www.dgi.polymtl.ca/vocab#> .
@prefix  prof:  <http://www.dgi.polymtl.ca/profs#> .

_:n1
    local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> ;
    local:worksAt  <http://www.dgi.polymtl.ca> ;
    local:name "Michel Gagnon" .

local:JeanPaul2 local:knows   _:n1 .
```

Sérialisation Turtle

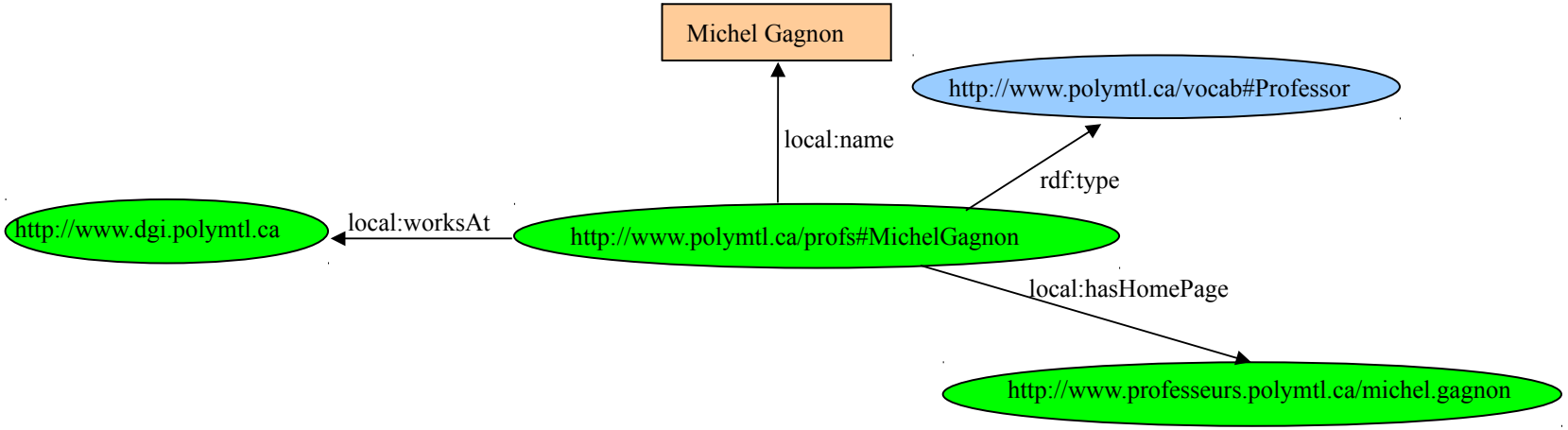



```
@prefix  local: <http://www.dgi.polymtl.ca/vocab#> .
@prefix  prof:  <http://www.dgi.polymtl.ca/profs#> .

local:JeanPaul2 local:knows
    [ local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> ;
      local:worksAt  <http://www.dgi.polymtl.ca> ;
      local:name "Michel Gagnon" ] .
```

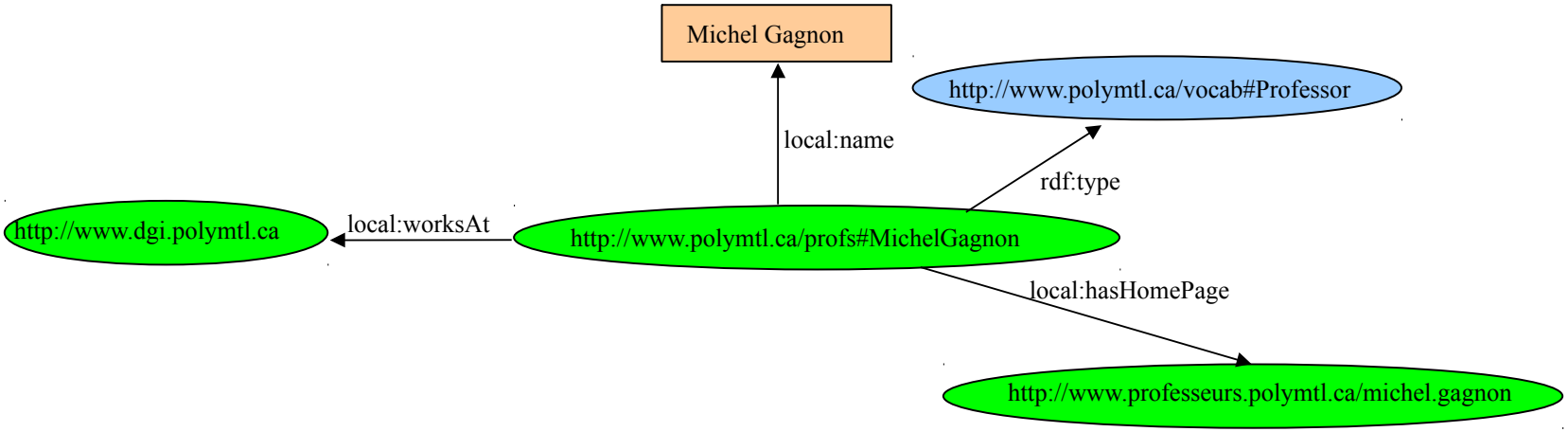
RDF – Déclaration de type de ressource

- Pour identifier le type d'une ressource:
  - Utiliser le prédicat rdf:type pré-défini par RDF
  - Remplacer la balise rdf:Description par le type de la ressource
- À noter qu'une ressource peut avoir plusieurs types
- En Turtle, on peut utiliser le prédicat a

Type – Exemple

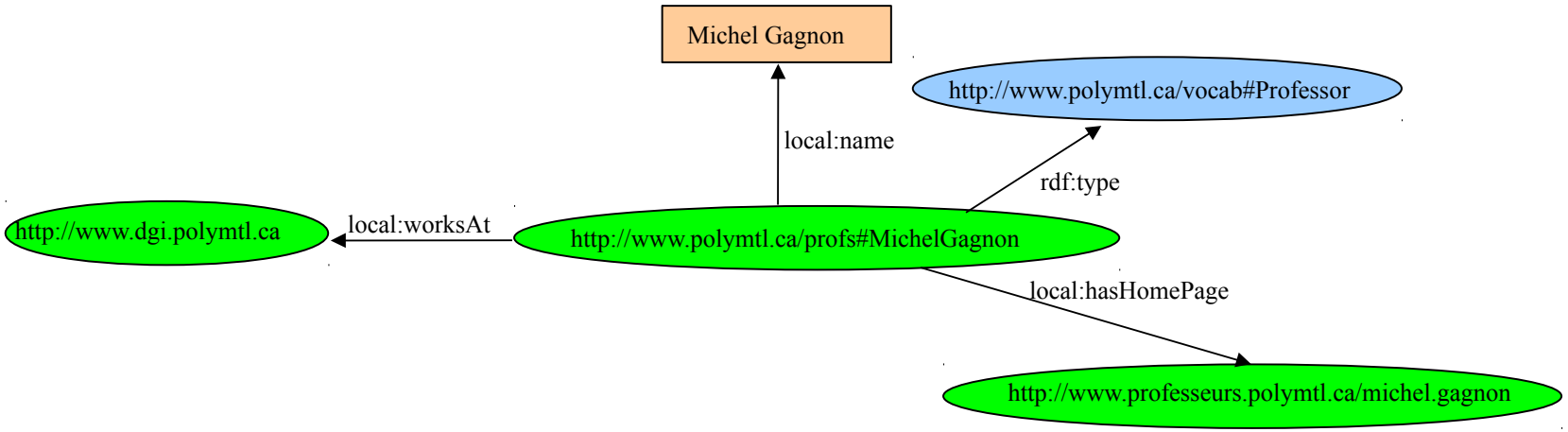


Type – Exemple - RDF/XML



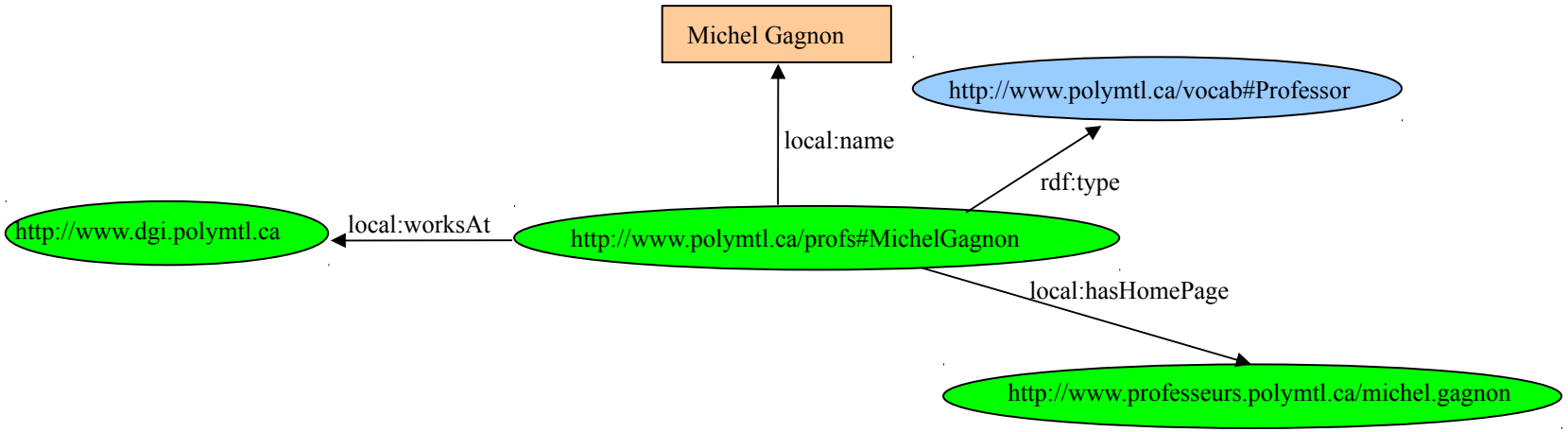

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:local="http://www.dgi.polymtl.ca/vocab#">
<rdf:Description rdf:about="http://www.polymtl.ca/profs#MichelGagnon">
    <rdf:type rdf:resource="http://www.polymtl.ca/vocab#Professor"/>
    <local:hasHomePage resource="http://www.professeurs.polymtl.ca/michel.gagnon"/>
    <local:worksAt resource="http://www.dgi.polymtl.ca"/>
    <local:name>Michel Gagnon</local:name>
</rdf:Description>
</rdf:RDF>
```

Type – Exemple - RDF/XML



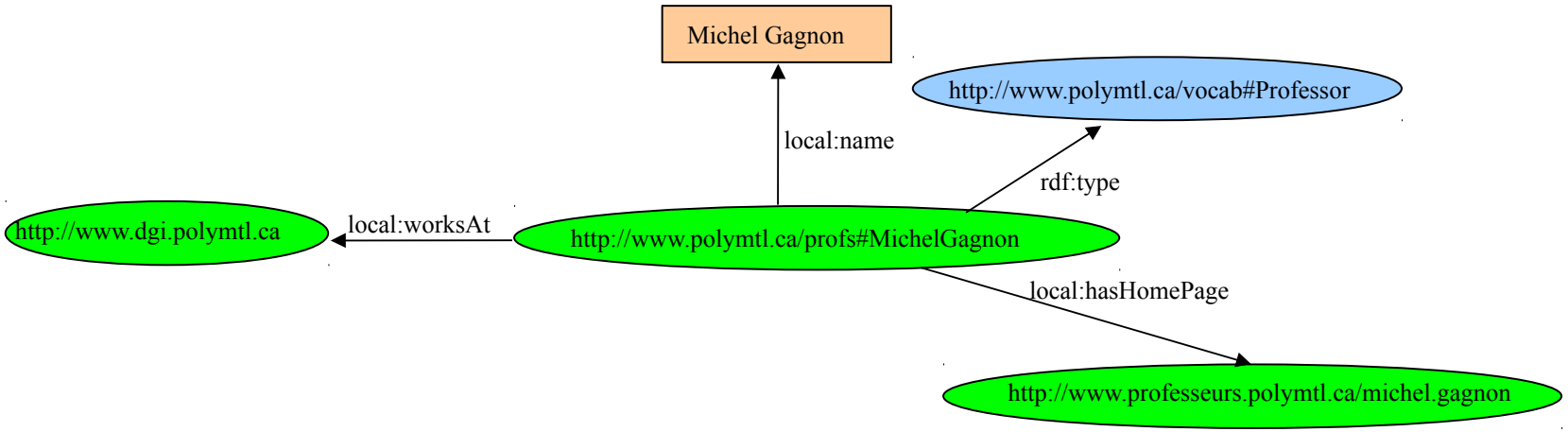

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:local="http://www.dgi.polymtl.ca/vocab#">
<local:Professor rdf:about="http://www.polymtl.ca/profs#MichelGagnon">
    <local:hasHomePage resource="http://www.professeurs.polymtl.ca/michel.gagnon"/>
    <local:worksAt resource="http://www.dgi.polymtl.ca"/>
    <local:name>Michel Gagnon</local:name>
</local:Professor>
</rdf:RDF>
```

Type – Exemple - Turtle




```
@prefix  local:  <http://www.dgi.polymtl.ca/vocab#> .
@prefix  prof:   <http://www.dgi.polymtl.ca/profs#> .

prof:MichelGagnon
        rdf:type   local:Professor ;
        local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> ;
        local:worksAt resource  <http://www.dgi.polymtl.ca> ;
        local:name  “Michel Gagnon” .
```

Type – Exemple - Turtle

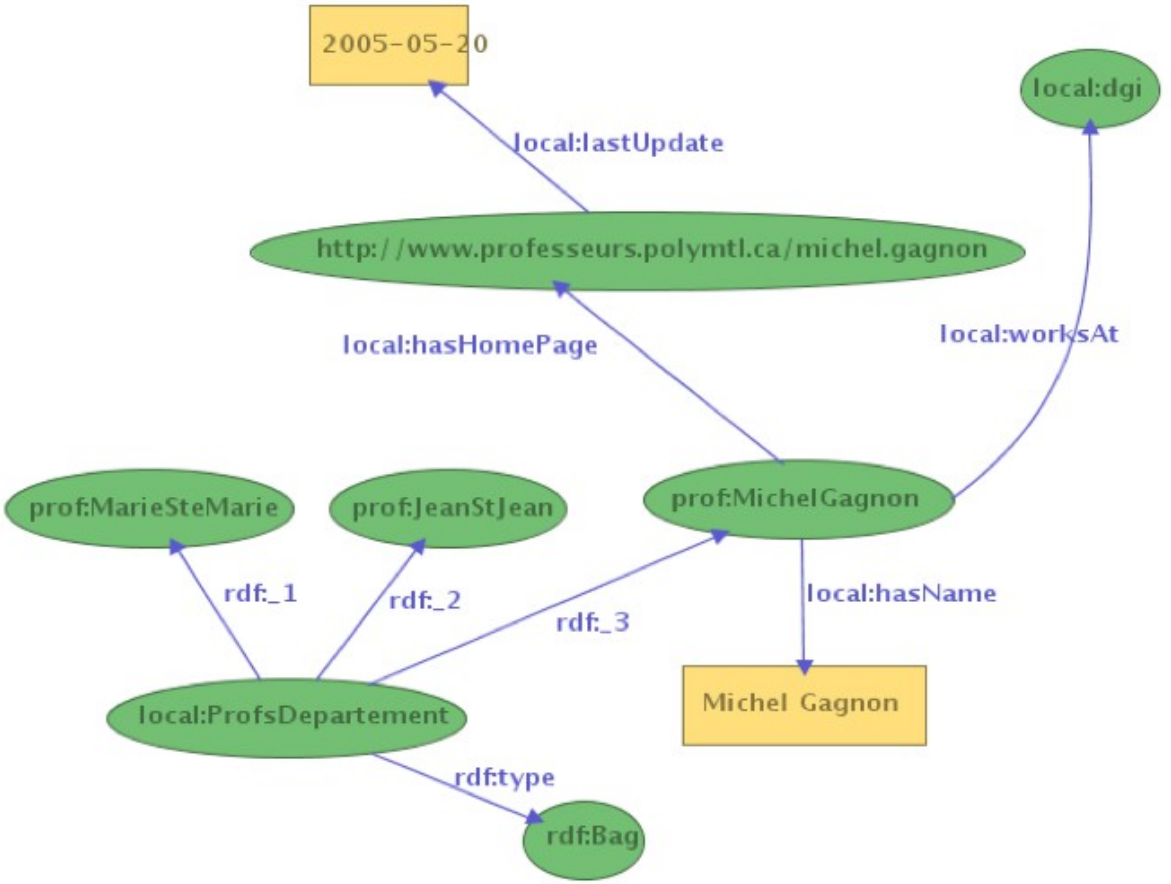



```
@prefix  local:  <http://www.dgi.polymtl.ca/vocab#> .
@prefix  prof:   <http://www.dgi.polymtl.ca/profs#> .

prof:MichelGagnon
        a   local:Professor ;
        local:hasHomePage  <http://www.professeurs.polymtl.ca/michel.gagnon> ;
        local:worksAt resource  <http://www.dgi.polymtl.ca> ;
        local:name  “Michel Gagnon” .
```

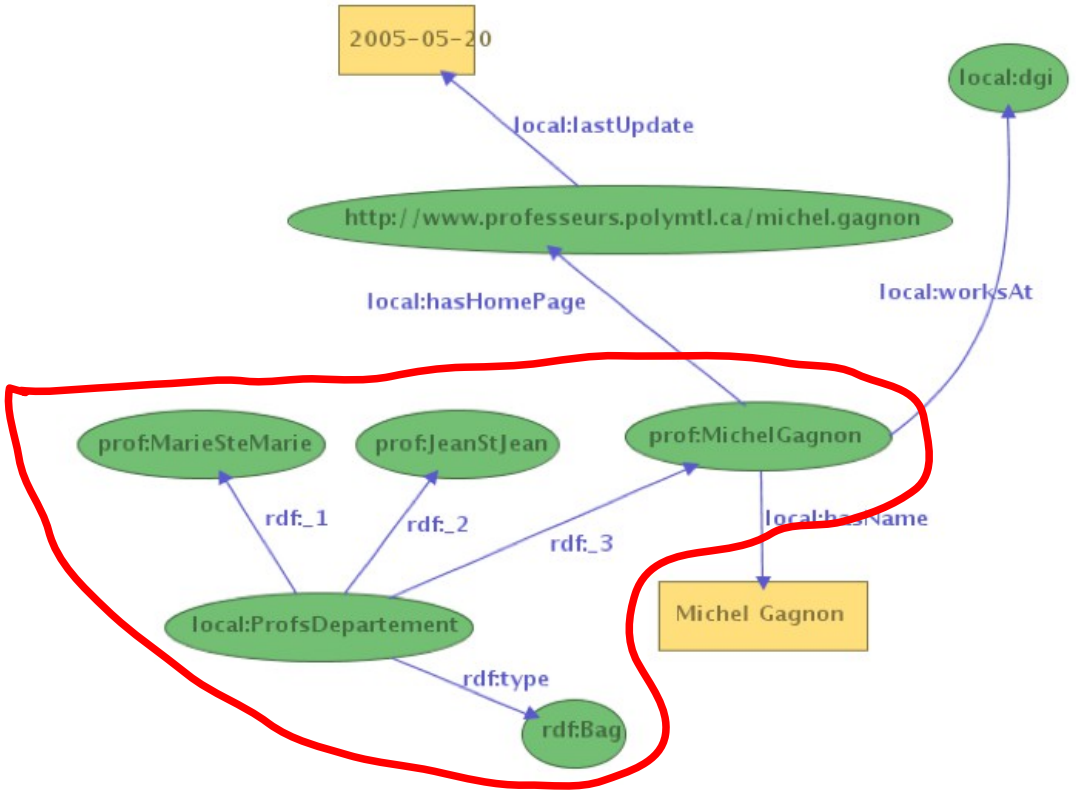
Conteneurs

- Ressource qui contient d'autres ressources
- Trois classes de conteneurs:
  - `rdf:Bag`: pas d'ordre entre les membres
    `rdf:Seq`: membres ordonnés
  - `ref:Alt`: on s'attend à ce qu'un seul des éléments soit sélectionné
- Conteneur relié à un membre par la relation `rdf:_n`
- Pas de contraintes sur la description de conteneurs

Conteneurs - Exemple

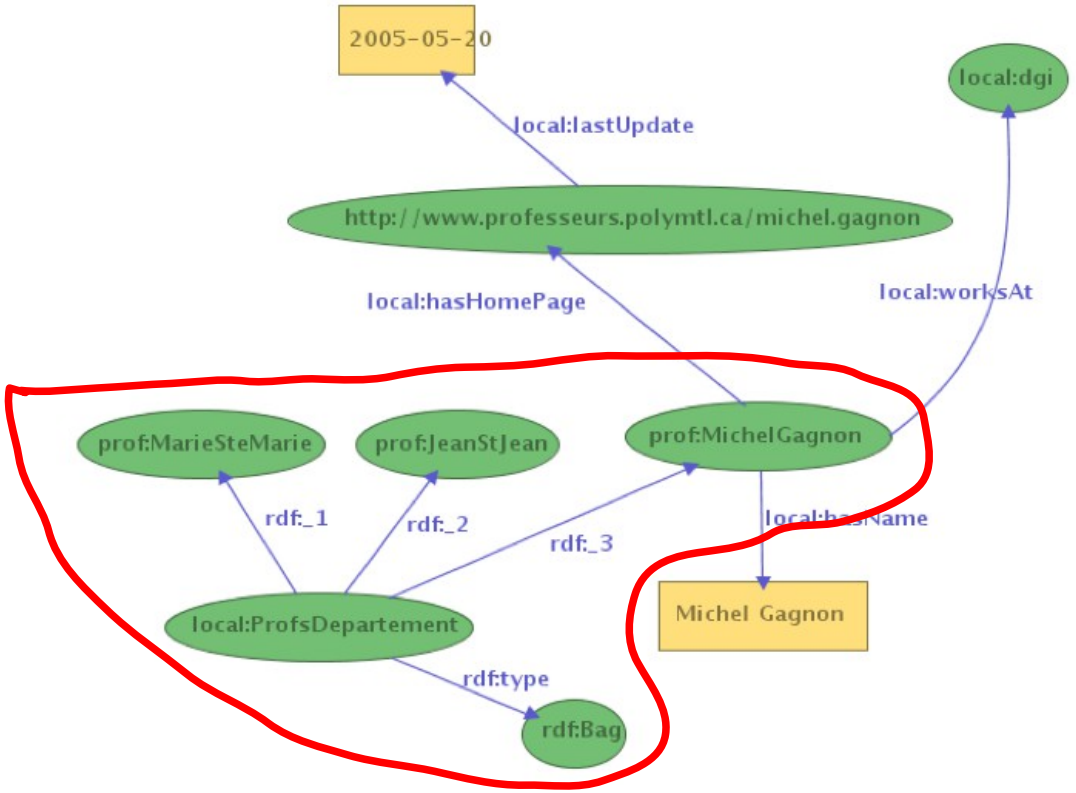


Conteneurs – Exemple – RDF/XML



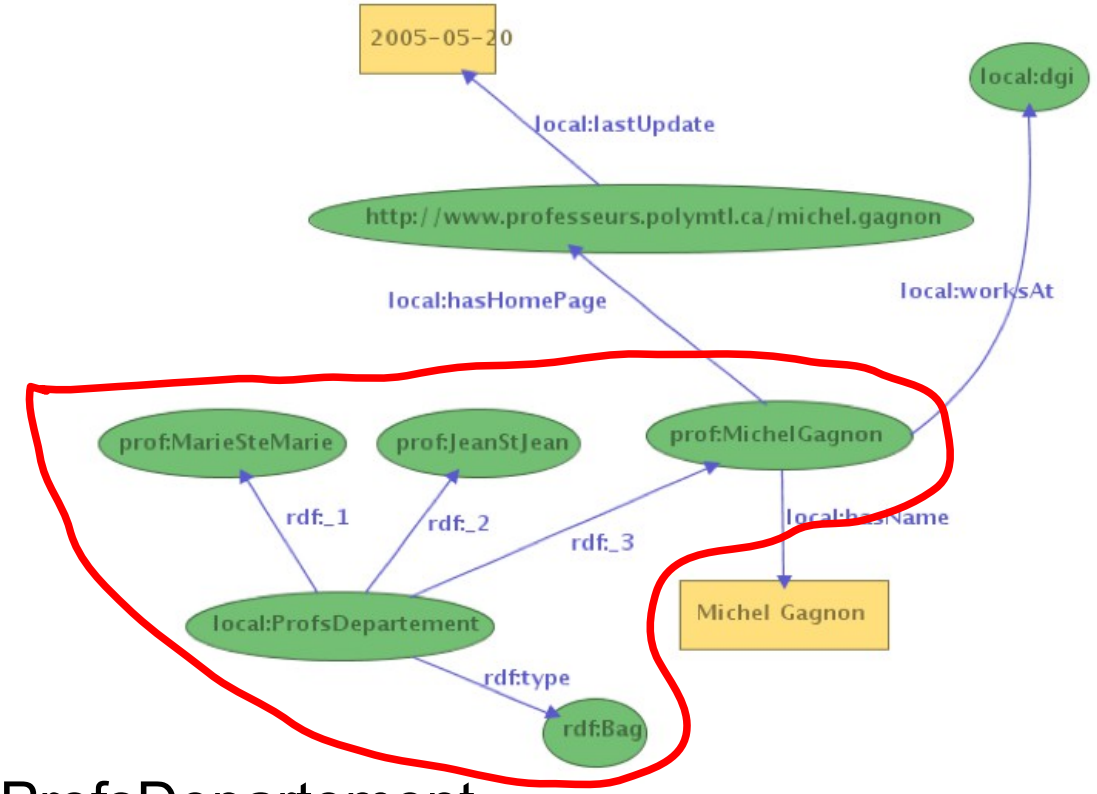

```
<rdf:Bag rdf:about="http://www.polymtl.ca/Vocabulary#ProfsDepartement">
   <rdf:_1 rdf:resource="http://www.polymtl.ca/Profs#MarieSteMarie"/>
   <rdf:_2 rdf:resource="http://www.polymtl.ca/Profs#JeanStJean"/>
   <rdf:_3 rdf:resource="http://www.polymtl.ca/Profs#MichelGagnon"/>
 </rdf:Bag>
```

Conteneurs – Exemple – RDF/XML




```
<rdf:Bag rdf:about="http://www.polymtl.ca/Vocabulary#ProfsDepartement">
   <rdf:li rdf:resource="http://www.polymtl.ca/Profs#MarieSteMarie"/>
   <rdf:li rdf:resource="http://www.polymtl.ca/Profs#JeanStJean"/>
   <rdf:li rdf:resource="http://www.polymtl.ca/Profs#MichelGagnon"/>
 </rdf:Bag>
```

Conteneurs – Exemple – Turtle

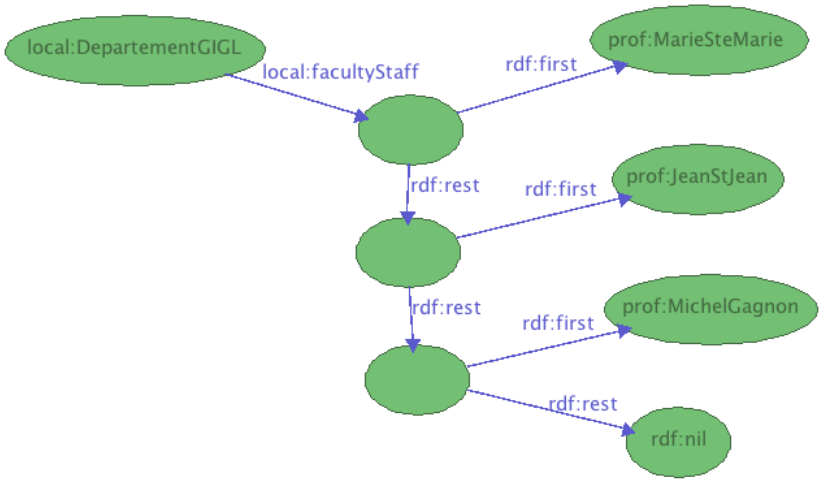



```
local:ProfsDepartement
      a rdf:Bag ;
      rdf:_1 prof:MarieSteMarie ;
      rdf:_2 prof:JeanStJean ;
      rdf:_3 prof:MichelGagnon .
```

Collections

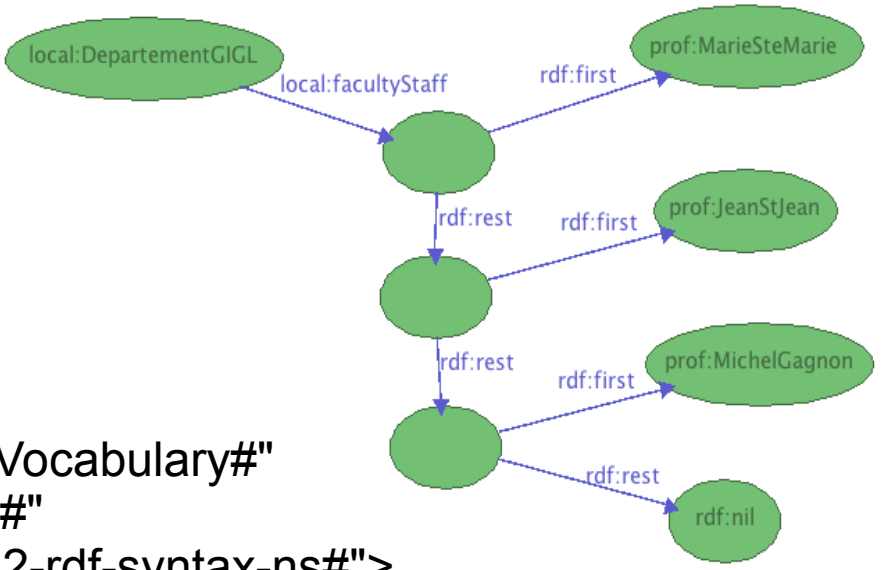
- Listes fermées, contrairement aux conteneurs
- Balise `rdf:nil` pour représenter la liste vide
- Liste construite de manière récursive, en utilisant les prédicats `rdf:first` et `rdf:rest`
- Il exise un forme abbrégée
- Il n'y a pas de contraintes sur l'utilisation des balises `rdf:first` et `rdf:rest`

Collections – Exemple – RDF/XML



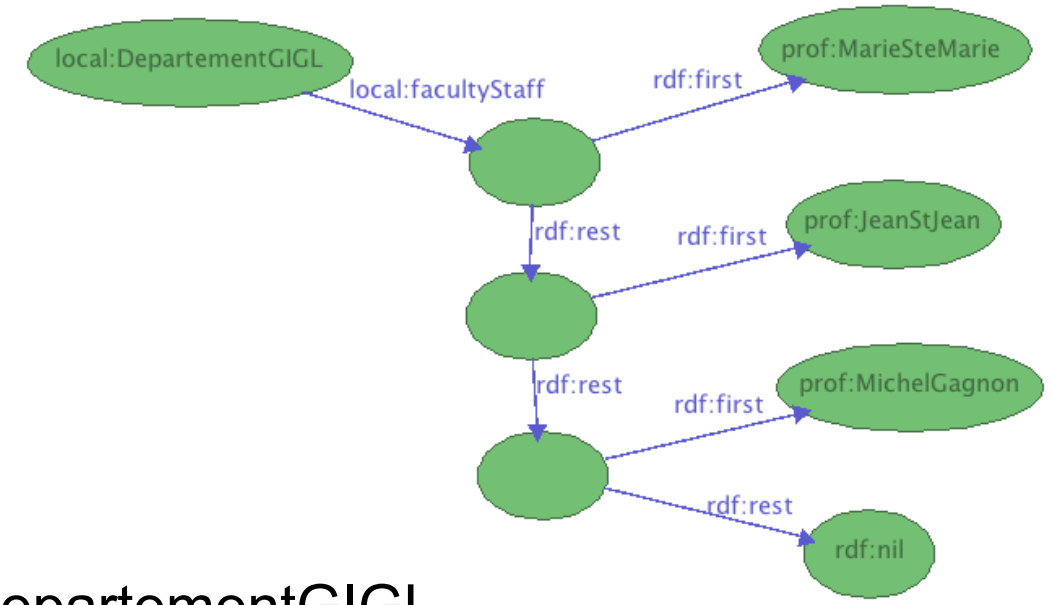

```
<rdf:RDF xmlns:local="http://www.polymtl.ca/Vocabulary#" xmlns:prof="http://www.polymtl.ca/Profs#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description rdf:about="http://www.polymtl.ca/Vocabulary#DepartementGIGL">
            <local:facultyStaff>
                  <rdf:Description>
                        <rdf:first rdf:resource="http://www.polymtl.ca/Profs#MarieSteMarie" />
                        <rdf:rest>
                              <rdf:Description>
                                    <rdf:first rdf:resource="http://www.polymtl.ca/Profs#JeanStJean" />
                                    <rdf:rest>
                                          <rdf:Description>
                                                <rdf:first rdf:resource="http://www.polymtl.ca/Profs#MichelGagnon" />
                                                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil" />
                                          </rdf:Description>
                                    </rdf:rest>
                              </rdf:Description>
                        </rdf:rest>
                  </rdf:Description>
            </local:facultyStaff>
      </rdf:Description>
</rdf:RDF>
```

Collections – Exemple – RDF/XML



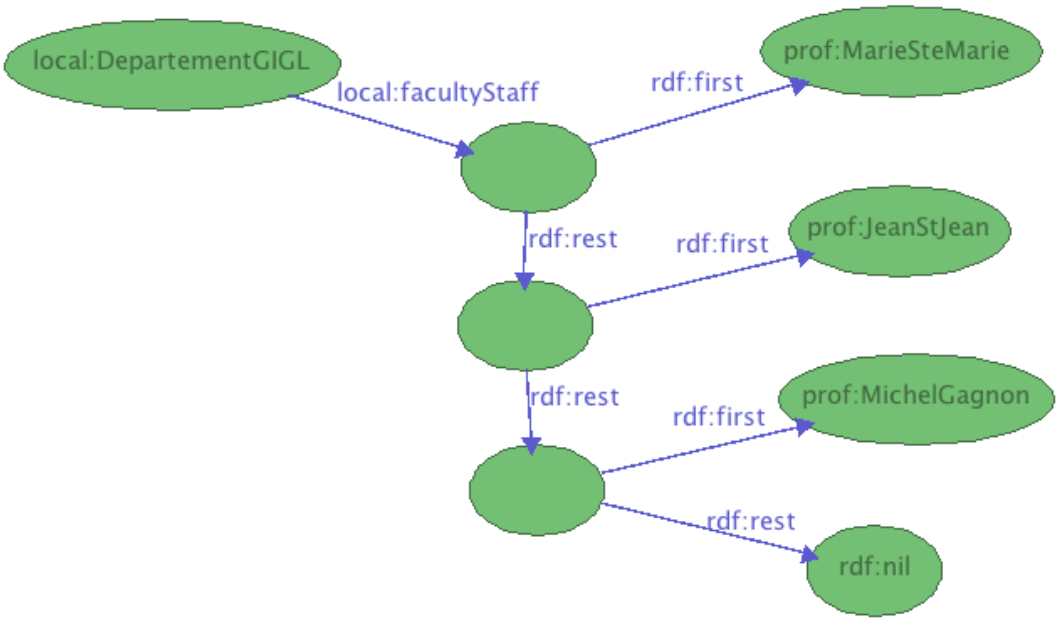

```
<rdf:RDF xmlns:local="http://www.polymtl.ca/Vocabulary#"
        xmlns:prof="http://www.polymtl.ca/Profs#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

 <rdf:Description rdf:about="http://www.polymtl.ca/Vocabulary#DepartementGIGL">
     <local:facultyStaff rdf:parseType="Collection">
         <rdf:Description rdf:about="http://www.polymtl.ca/Profs#MarieSteMarie"/>
         <rdf:Description rdf:about="http://www.polymtl.ca/Profs#JeanStJean"/>
         <rdf:Description rdf:about="http://www.polymtl.ca/Profs#MichelGagnon"/>
     </local:facultyStaff>
  </rdf:Description>
</rdf:RDF>
```

Collections – Exemple – Turtle




```
local:DepartementGIGL
    local:facultyStaff
        [ a rdf:List ;
          rdf:first prof:MarieSteMarie ;
          rdf:rest [ rdf:first prof:JeanStJean ;
                     rdf:rest  [ rdf:first prof:MichelGagnon ;
                                 rdf:rest rdf:nil ] ] ] .
```

Collections – Exemple – Turtle




```
local:DepartementGIGL
    local:facultyStaff
        ( prof:MarieSteMarie prof:JeanStJean prof:MichelGagnon ) .
```

Valeurs structurées

- Une valeur peut aussi avoir d'autres caractéristiques, comme une unité
- Exemple :

```
prod:item10245   exterms:weight   "2.4"^^xsd:decimal .
```

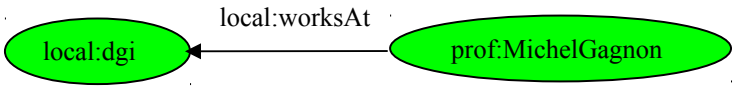
- Comment représenter les unités?
- En utilisant le prédicat `rdf:valeur`

```
prod:item10245   exterms:weight
      [ rdf:value "2.4"^^xsd:decimal;
        exterms:units  exunits:kilograms ] .
```

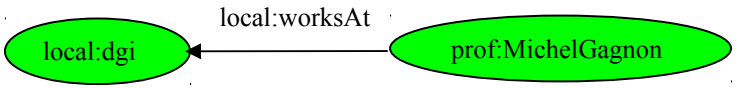
Réification

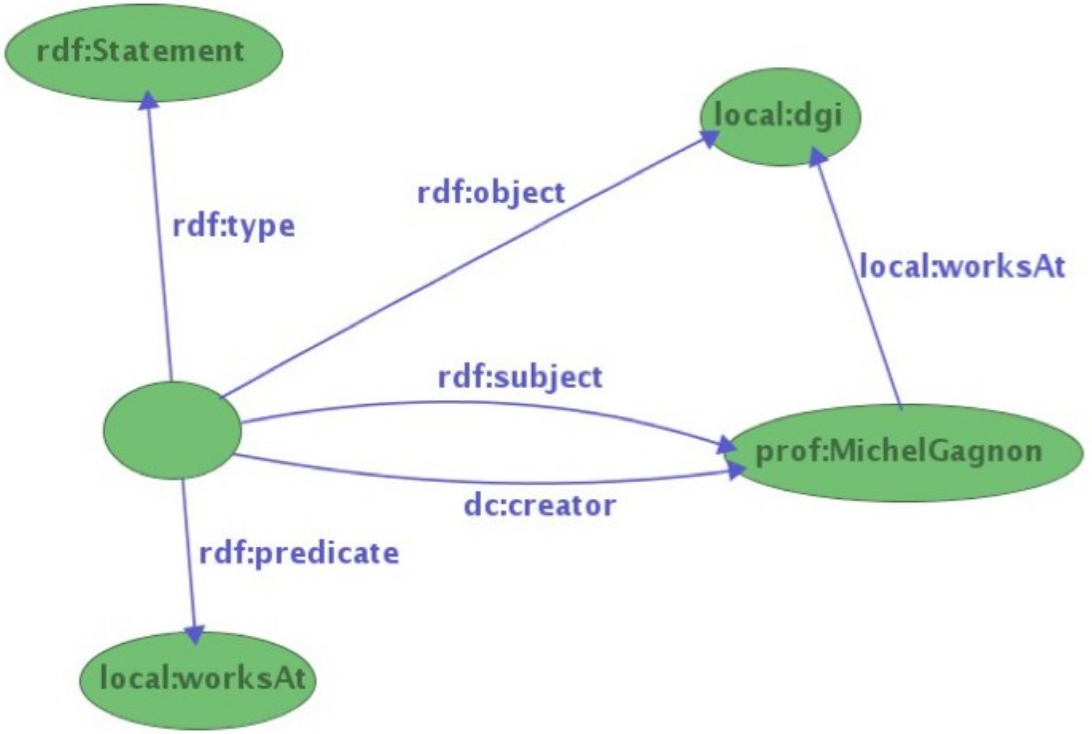
- La balise `rdf:Statement` permet de désigner une ressource qui est en fait un énoncé RDF
- Avec les prédicats `rdf:subject`, `rdf:predicat` et `rdf:object`, on peut indiquer les trois composantes de l'énoncé
- RDF n'offre pas de moyen pour spécifier l'énoncé RDF qui correspond à la réification
- En ajoutant l'attribut `rdf:ID` à une propriété (en notation RDF/XML), on spécifie implicitement une réification de l'énoncé concerné par cette propriété

Réification - Exemple

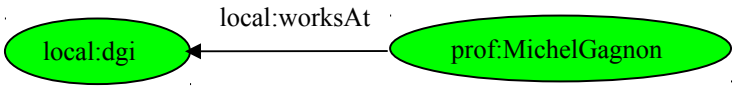


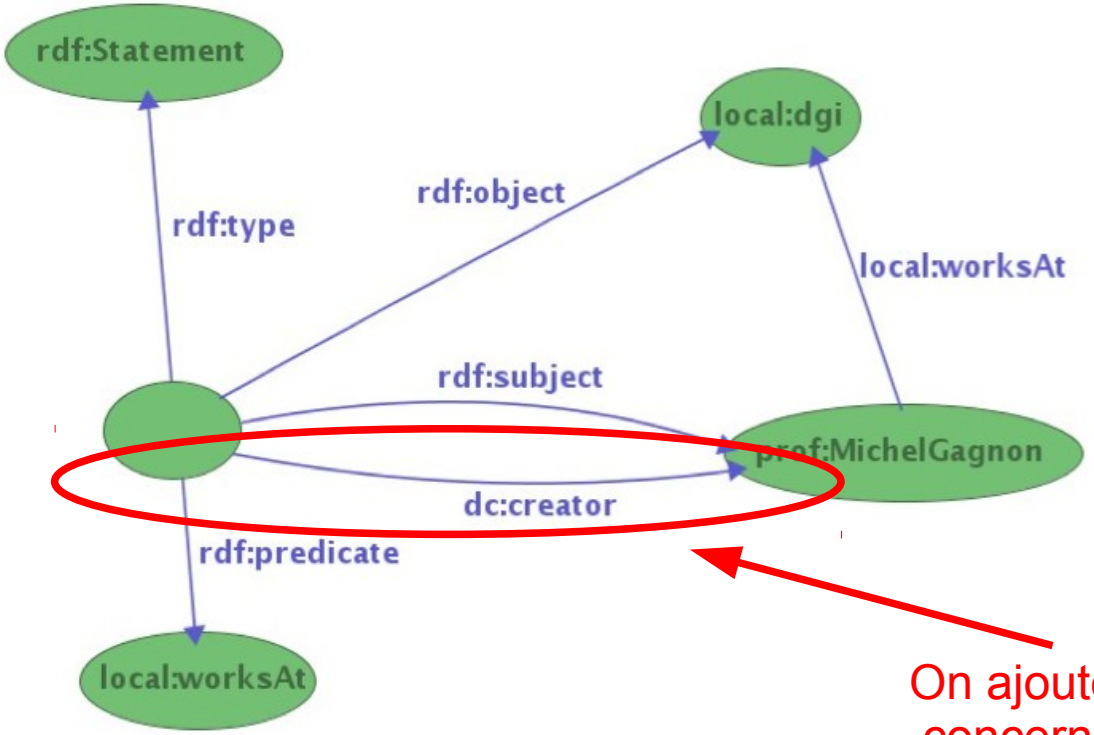
Réification - Exemple





Réification - Exemple



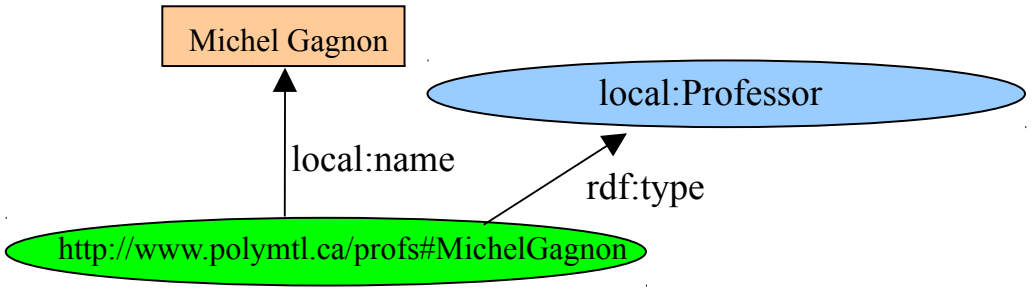


On ajoute des informations concernant l'énoncé

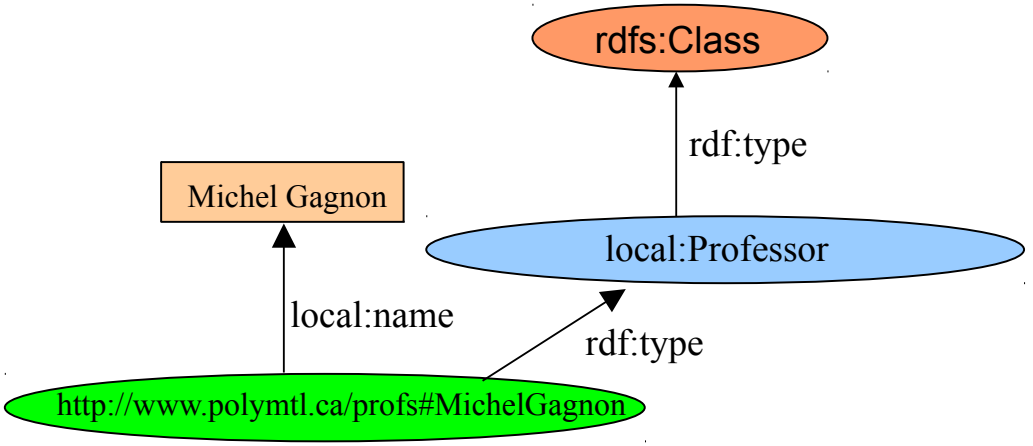
RDF Schema - Classes

- Une ressource peut appartenir à plus d'une classe
- Un type appartient à la classe `rdfs:Class`
- RDFS permet de définir une hiérarchie de classes, grâce au prédicat `rdfs:subClassOf`

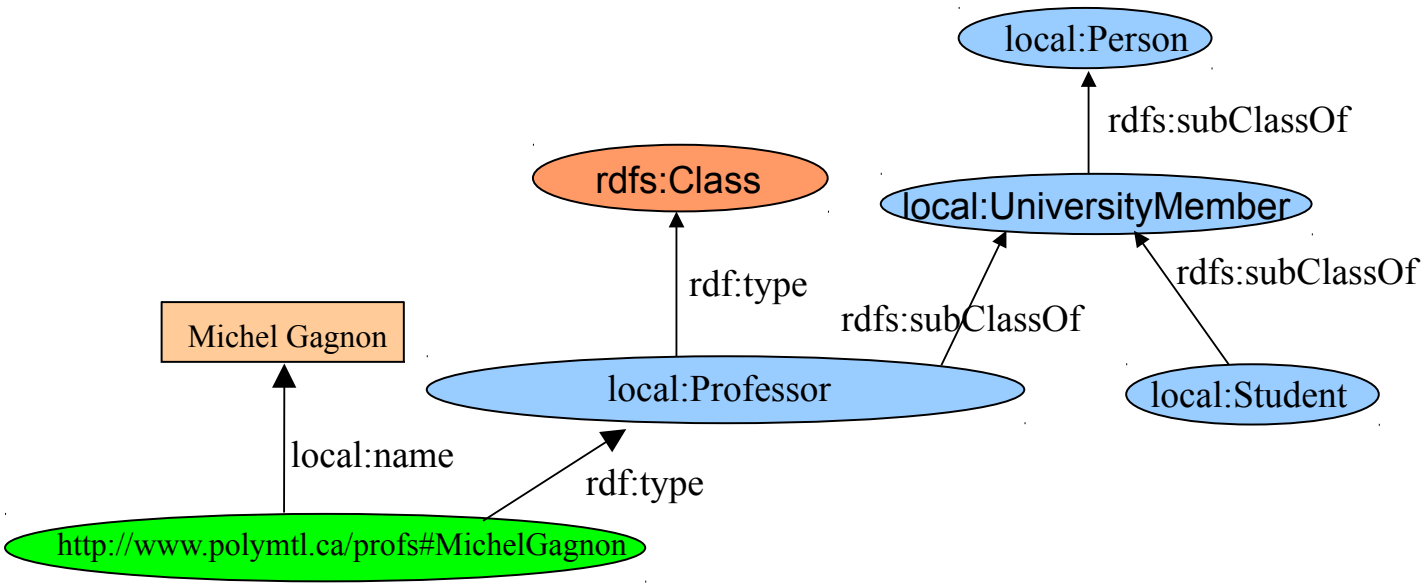
RDF Schema - Classes



RDF Schema - Classes



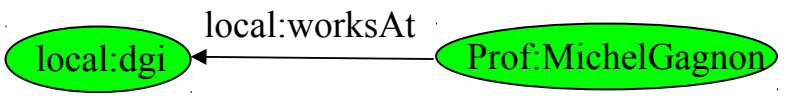
RDF Schema - Classes

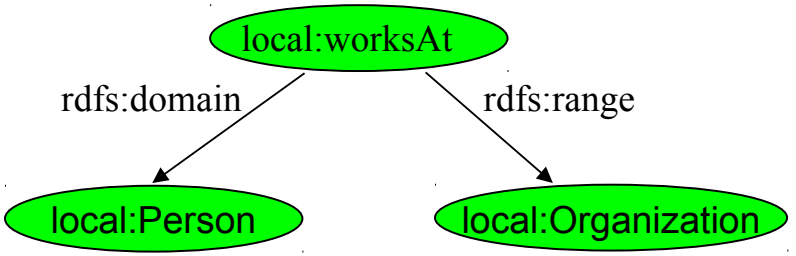


RDF Schema – Propriétés

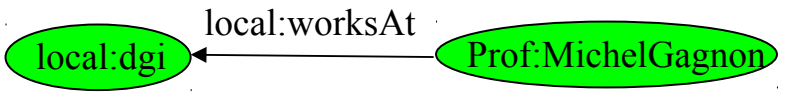
- Toutes les propriétés ont pour type la classe `rdf:Property`
- On peut établir des hiérarchies de propriétés, grâce au prédicat `rdfs:subPropertyOf`
- On peut définir le domaine et l'image d'une propriété, en utilisant les prédicats `rdfs:domain` et `rdfs:range`, respectivement
- Les propriétés sont globales (on peut donc y ajouter des informations n'importe où)

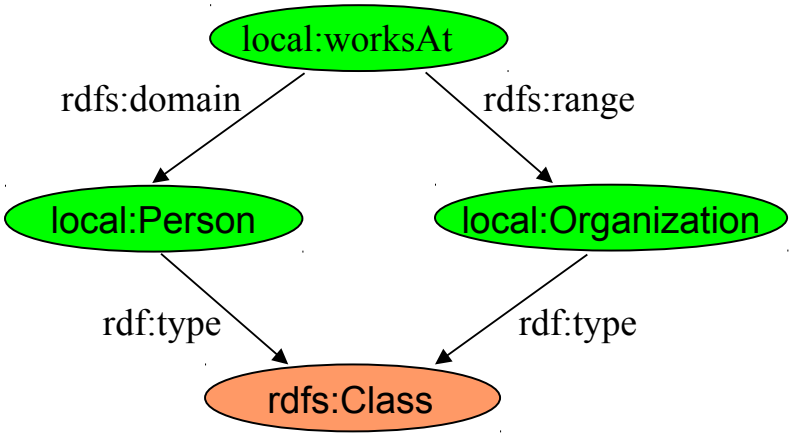
RDF Schema – Propriétés



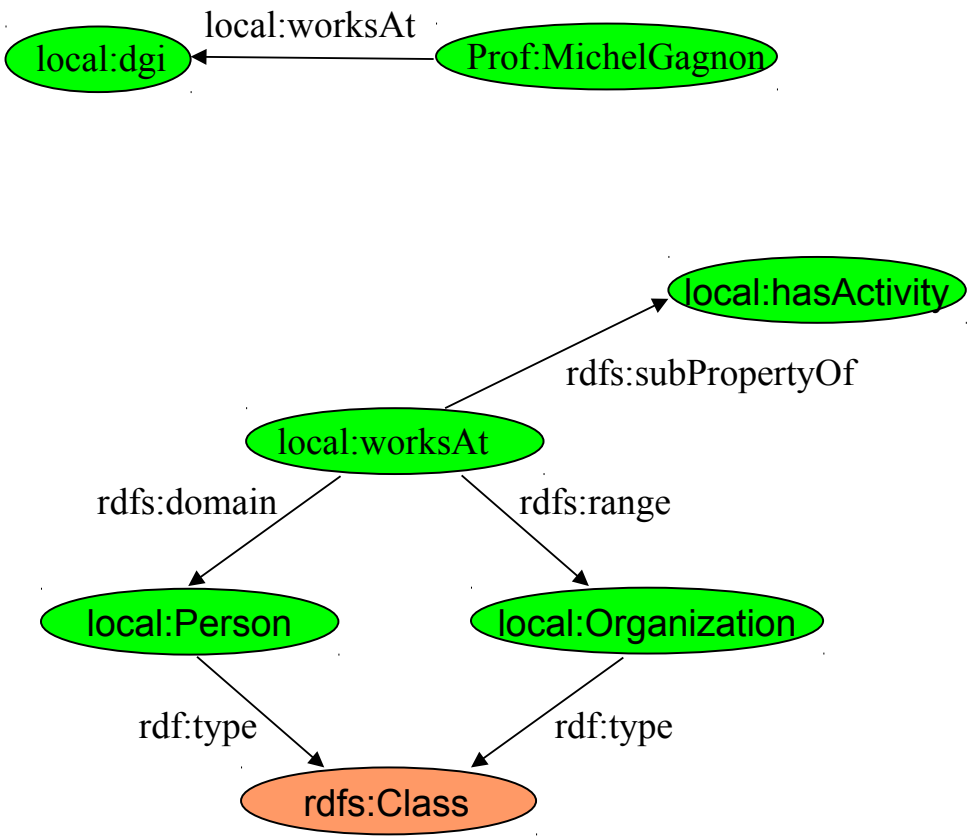


RDF Schema – Propriétés



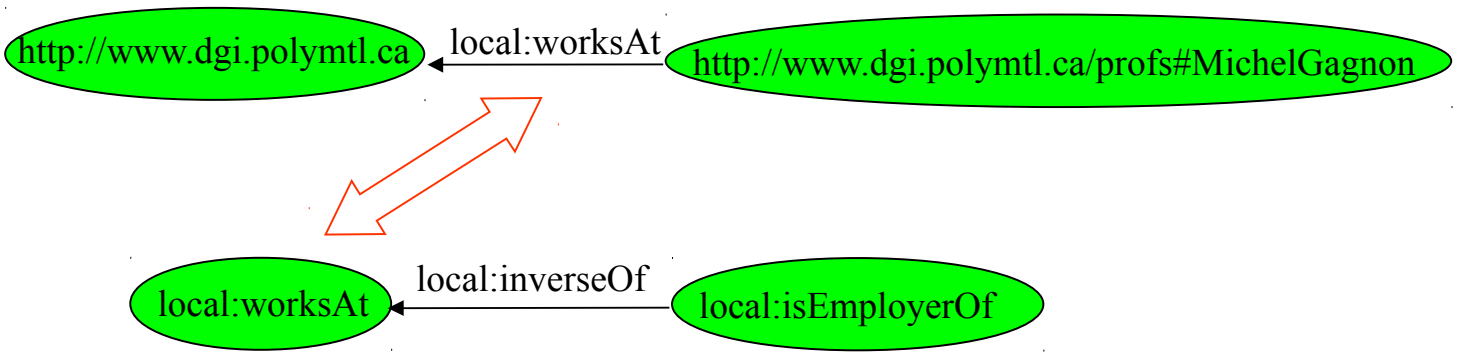


RDF Schema – Propriétés



RDF Schema – Propriétés

- En RDF tout est une ressource, même les propriétés
- Ceci signifie qu'on peut ajouter des descriptions aux propriétés:



Exercice – Dessinez le graphe :

```
<rdf:RDF
   xmlns="http://www.polymtl.ca#"
   xml:base="http://www.polymtl.ca#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
<A rdf:ID="x1">
   <p rdf:resource= "http://www.polymtl.ca"/>
</A>
</rdf:RDF>
```

Exercice – Dessinez le graphe :

```
<rdf:RDF
   xmlns="http://www.polymtl.ca#"
   xml:base="http://www.polymtl.ca#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
<A rdf:ID="x1">
   <p>
        <rdf:Description>
             <q rdf:resource= "http://www.polymtl.ca"/>
        </rdf:Description>
   </p>
</A>
</rdf:RDF>
```

Exercice – Dessinez le graphe :

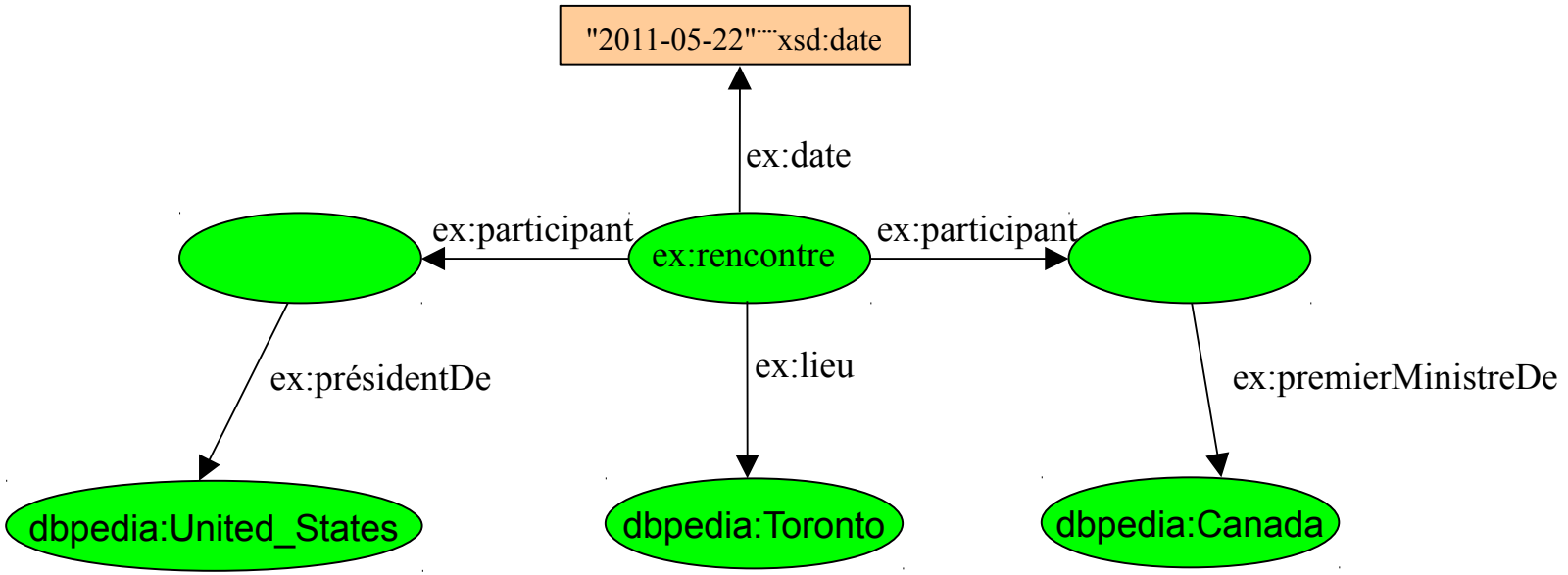
```
<?xml version="1.0"?>
<rdf:RDF
   xmlns="http://www.polymtl.ca#"
   xml:base="http://www.polymtl.ca#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
<A rdf:ID="x1">
   <p rdf:parseType="Collection">
      <rdf:Description rdf:ID="x2"/>
      <rdf:Description>
          <q rdf:resource="x3"/>
      </rdf:Description>
   </p>
</A>
</rdf:RDF>
```

Exercice – Représenter en RDF :

Le président des États-Unis a rencontré le premier ministre du Canada à Toronto le 22 mai 2011.

Exercice – Représenter en RDF :

Le président des États-Unis a rencontré le premier ministre du Canada à Toronto le 22 mai 2011.



Exercice – Représenter en RDF :

- Denyse Baillargeon, auteure d'un article citée dans Érudit (« La crise ordinaire: Les ménagères montréalaises et la crise des années trente »), est professeure titulaire au département d'histoire de l'U. de M., et membre du Groupe d'histoire de Montréal
- Elle apparaît dans un vidéo du CÉRIUM, avec d'autres intervenants : Les Khmers Rouges / Le féminisme : au Québec, en Inde et aux USA
- Elle a publié un article d'opinion dans Le Devoir le 14 octobre 2011 : « Le soi-disant déclin de l'histoire nationale au Québec »
- Le 21 octobre, toujours dans Le Devoir, Robert Comeau, professeur associé du département d'histoire de l'UQAM, a publié une réplique à son article : « Réplique à Denyse Baillargeon - Le déni de l’histoire nationale du Québec »

Fusion de graphes RDF

- Avant de fusionner deux graphes RDF, il faut renommer les noeuds vides de manière à ce qu'il n'y ait aucun identificateur de noeud vide commun aux deux graphes
- Une fois ceci fait, on effectue l'union des deux graphes

Fusion - Exemple

```
ex:a  ex:prop1  _:n1  .
ex:a  ex:prop2  _:n2  .
```

+

```
_:n1  ex:prop3   ex:c  .
ex:a  ex:prop3  _:n1  .
```

Fusion - Exemple

```
ex:a  ex:prop1  _:n3  .
ex:a  ex:prop2  _:n2  .
```

+

```
_:n1  ex:prop3   ex:c  .
ex:a  ex:prop3  _:n1  .
```

Fusion - Exemple

```
ex:a  ex:prop1  _:n3  .
ex:a  ex:prop2  _:n2  .
```

+

```
_:n1  ex:prop3   ex:c  .
ex:a  ex:prop3  _:n1  .
```

```
ex:a  ex:prop1  _:n3  .
ex:a  ex:prop2  _:n2  .
_:n1  ex:prop3   ex:c  .
ex:a  ex:prop3  _:n1  .
```

Fusion - Exemple

```
ex:a  ex:prop1  _:n3  .
ex:a  ex:prop2  _:n2  .
```

+

```
_:n1  ex:prop3   ex:c  .
ex:a  ex:prop3  _:n1  .
```

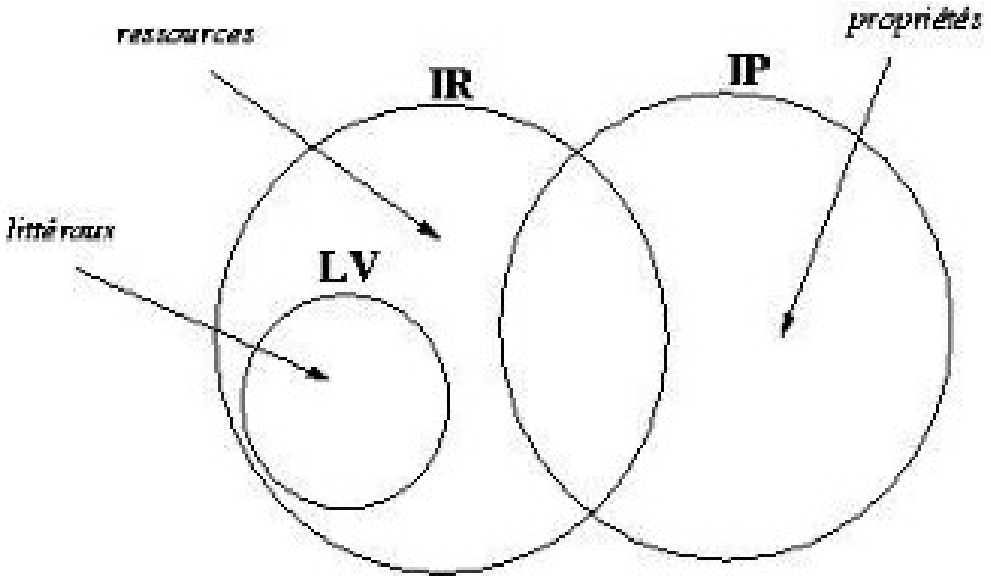
```
ex:a
    ex:prop1  [ ]  ;
    ex:prop2  [ ]  ;
    ex:prop3 [ ex:prop3 ex:c ]  .
```

Sémantique

- *Interprétation simple*: ne tient aucunement compte du vocabulaire utilisé
- *Rdf-interprétation*: tient compte du vocabulaire RDF (`rdf:type`, `rdf:Property`, `rdf:Bag`, etc.)
- *Rdfs-interprétation*: tient compte du vocabulaire RDFS (`rdfs:Class`, `rdfs:subClassOf`, etc.)

Interprétation simple

- I_{EXT} : $IP \rightarrow IR \times IR$
- IS : URI $\rightarrow$ $IR \cup IP$
- IL : $L \rightarrow IR$



Interprétation d'un graphe I(E)

- SI E ne contient aucun nœud vide :
  - I(E) = *faux* si I(E') est faux pour au moins un triplet E' contenu dans E. Sinon E(I) = *vrai*
  - Pour un triplet E = s p o . on a I(E) = *vrai* si I(p) $\in$ IP et $\langle s, o \rangle \in I_{EXT}(I(p))$. Sinon I(E) = *faux*
  - Si E est une URI, alors I(E) = IS(E)
  - Si E est un littéral simple "aaa" alors I(E) = aaa
  - Si E est un littéral simple "aaa"@ll, où ll spécifie la lange, alors I(E) = $\langle$aaa,ll$\rangle$
  - Si E est un littéral typé, alors I(E) = IL(E)

Interprétation - nœuds vides

- On utilise une fonction A qui associe chaque nœud vide à une ressource de l'ensemble IR
- Avant d'interpréter un graphe E, on va appliquer la fonction A à tous les nœuds vides qu'il contient. Appelons E' ce nouveau graphe
- Une fois ceci fait, on détermine I(E')
- I(E) est vrai s'il existe au moins une fonction A qui rendra I(E') = *vrai*

Rdf-interprétation

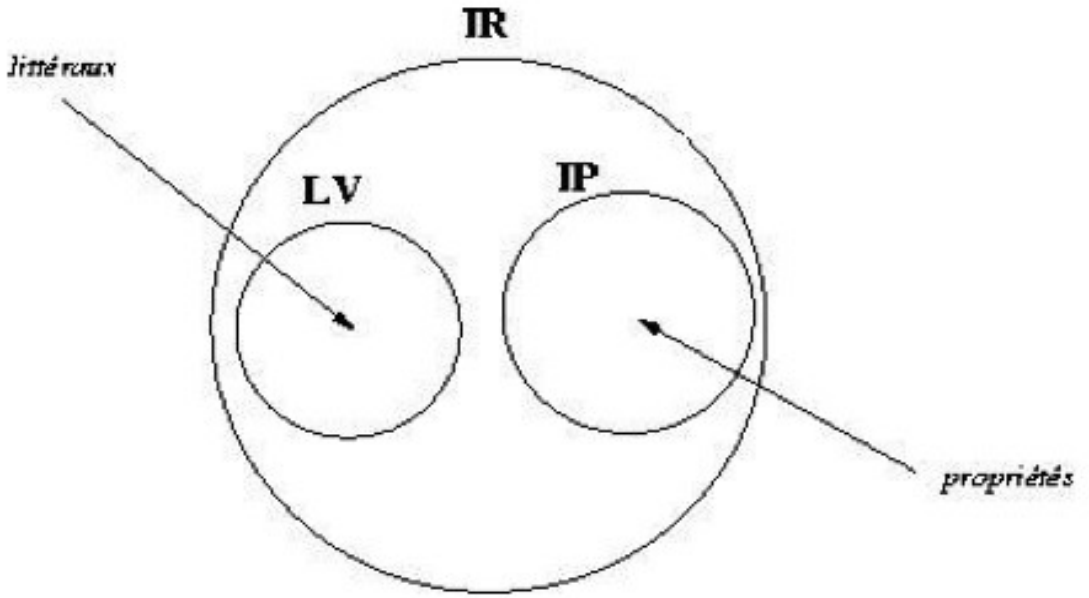
- Soit rdfV le vocabulaire RDF
- Une rdf-interprétation d'un vocabulaire V est une interprétation simple de V ∪ rdfV, qui respecte les conditions suivantes:
  - Deux conditions sur le terme rdf:XMLLiteral
  - Soit une entité x de l'univers. X ∈ IP si et seulement si $\langle x, I(\texttt{rdf:Property})\rangle \in I_{EXT}(I(\texttt{rdf:type}))$

Rdf-interprétation (suite)

- Axiomes :

```
rdf:type rdf:type rdf:Property .
rdf:subject rdf:type rdf:Property .
rdf:predicate rdf:type rdf:Property .
rdf:object rdf:type rdf:Property .
rdf:first rdf:type rdf:Property .
rdf:rest rdf:type rdf:Property .
rdf:value rdf:type rdf:Property .
rdf:_1 rdf:type rdf:Property .
rdf:_2 rdf:type rdf:Property .
...
rdf:nil rf:type rdf:List .
```

Rdf-interprétation



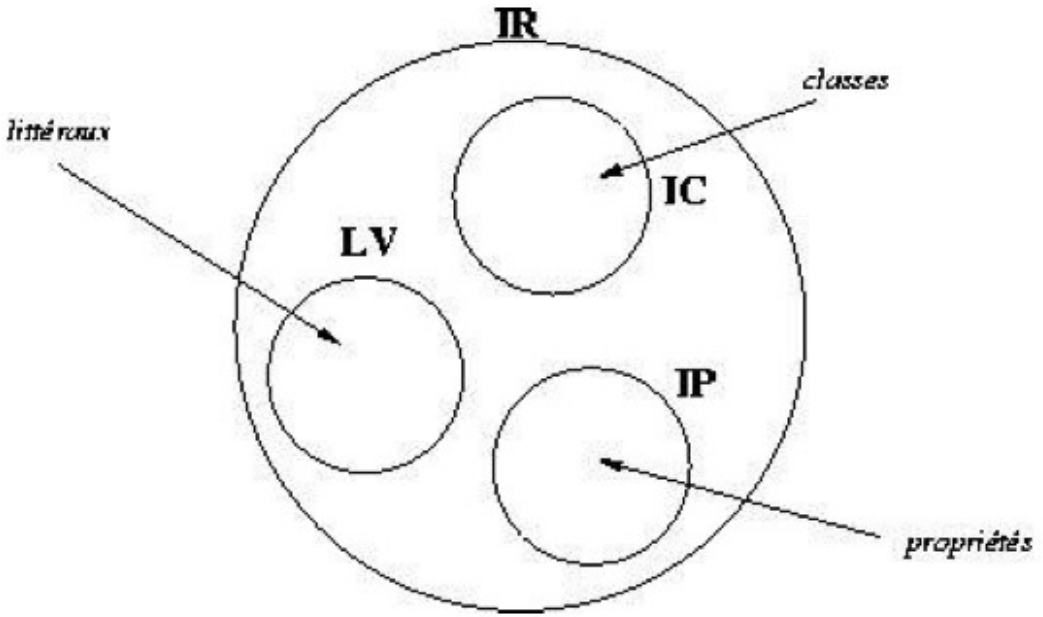
Rdfs-interprétation

- Une rdf-interprétation qui satisfait les conditions suivantes:
  - $x \in IC_{EXT}(y)$ si et seulement si $\langle x,y \rangle \in I_{EXT}(I(\texttt{rdf:type}))$
  - $IC = IC_{EXT}(I(\texttt{rdf:Class}))$
  - $IR = IC_{EXT}(I(\texttt{rdf:Resource}))$
  - $LV = IC_{EXT}(I(\texttt{rdf:Literal}))$
  - Autres conditions sur le vocabulaire RDFS

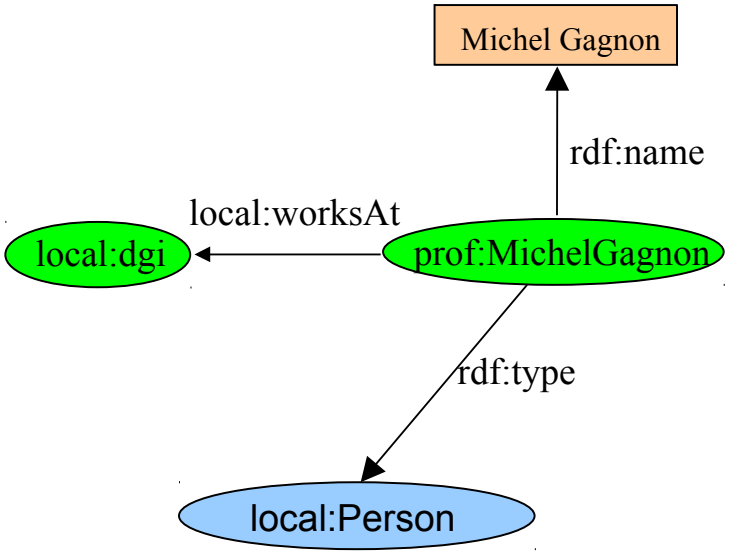
Rdfs-interprétation (suite)

- La définition de rdfs-interprétation implique les énoncés suivants:
  - `rdfs:Resource rdf:type rdfs:Class`
  - `rdfs:Class rdf:type rdfs:Class`
  - `rdfs:Literal rdf:type rdfs:Class`
  - `rdfs:domain rdf:type rdfs:Property`

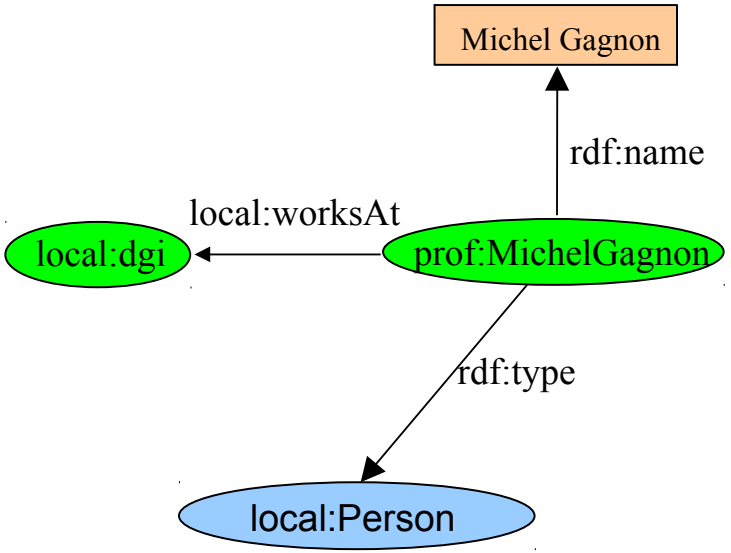
Rdfs-interprétation (suite)

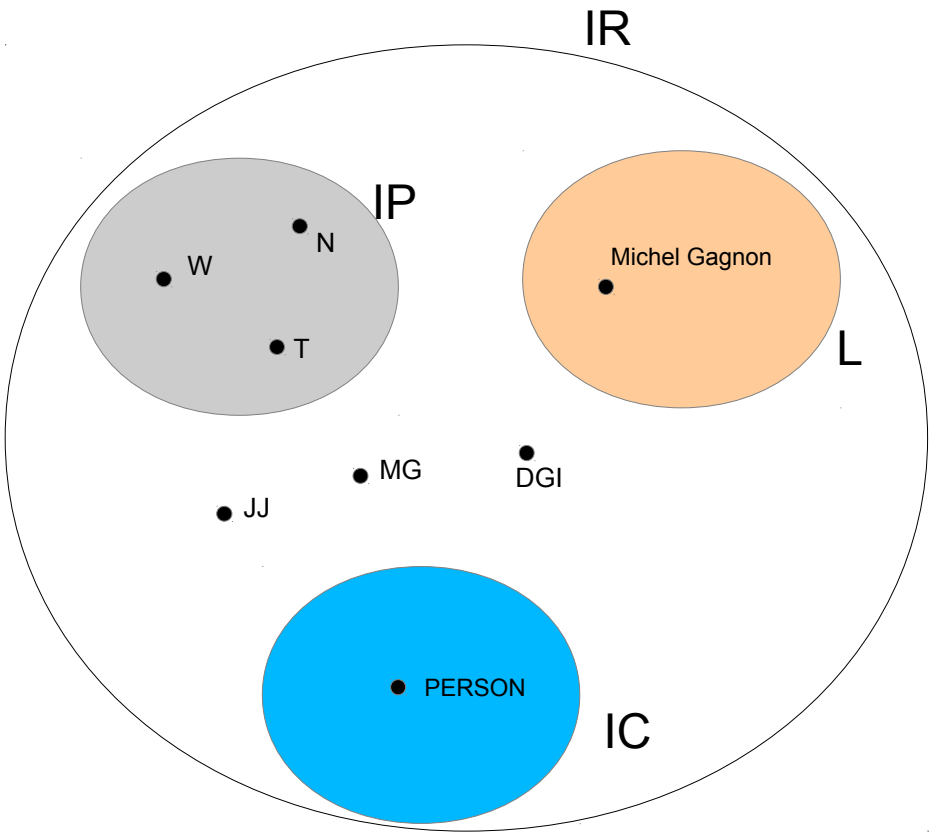


Sémantique - Exemple



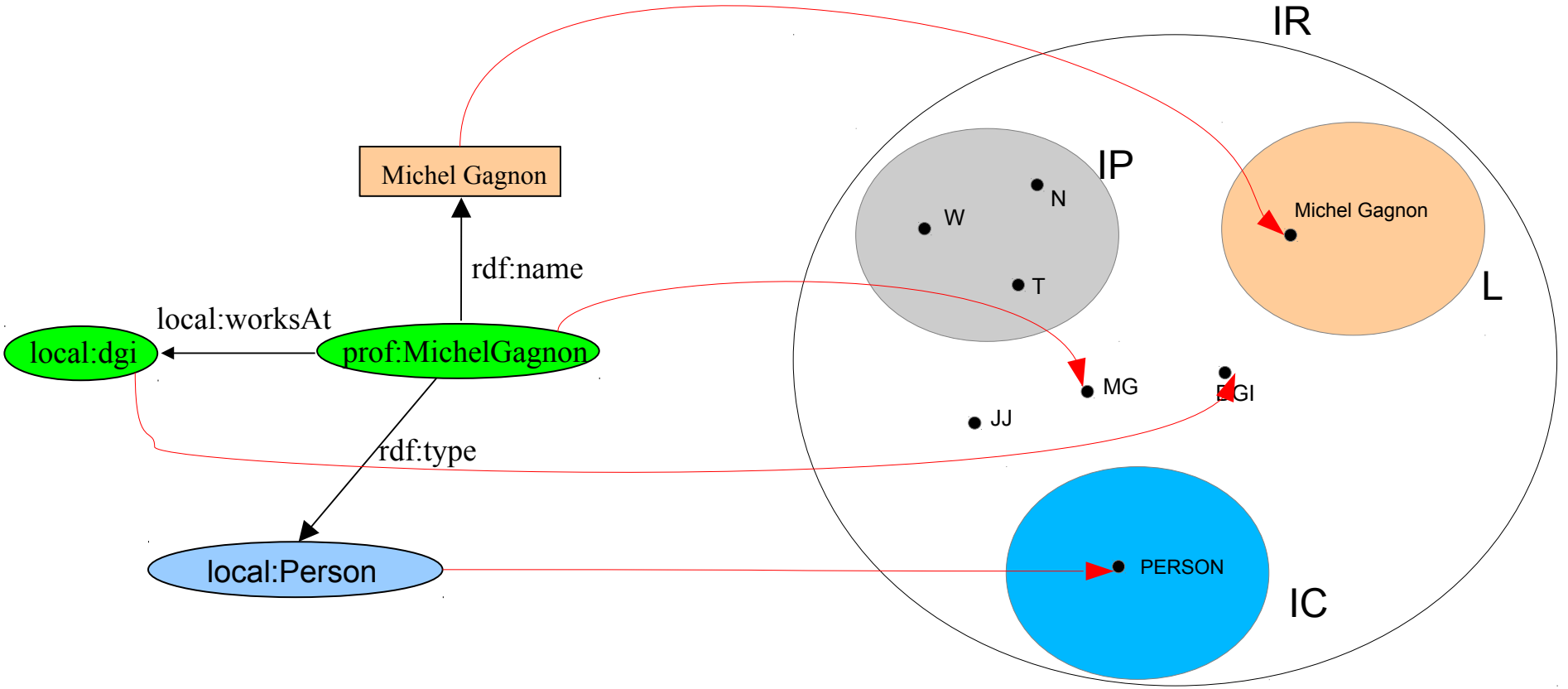
Sémantique - Exemple





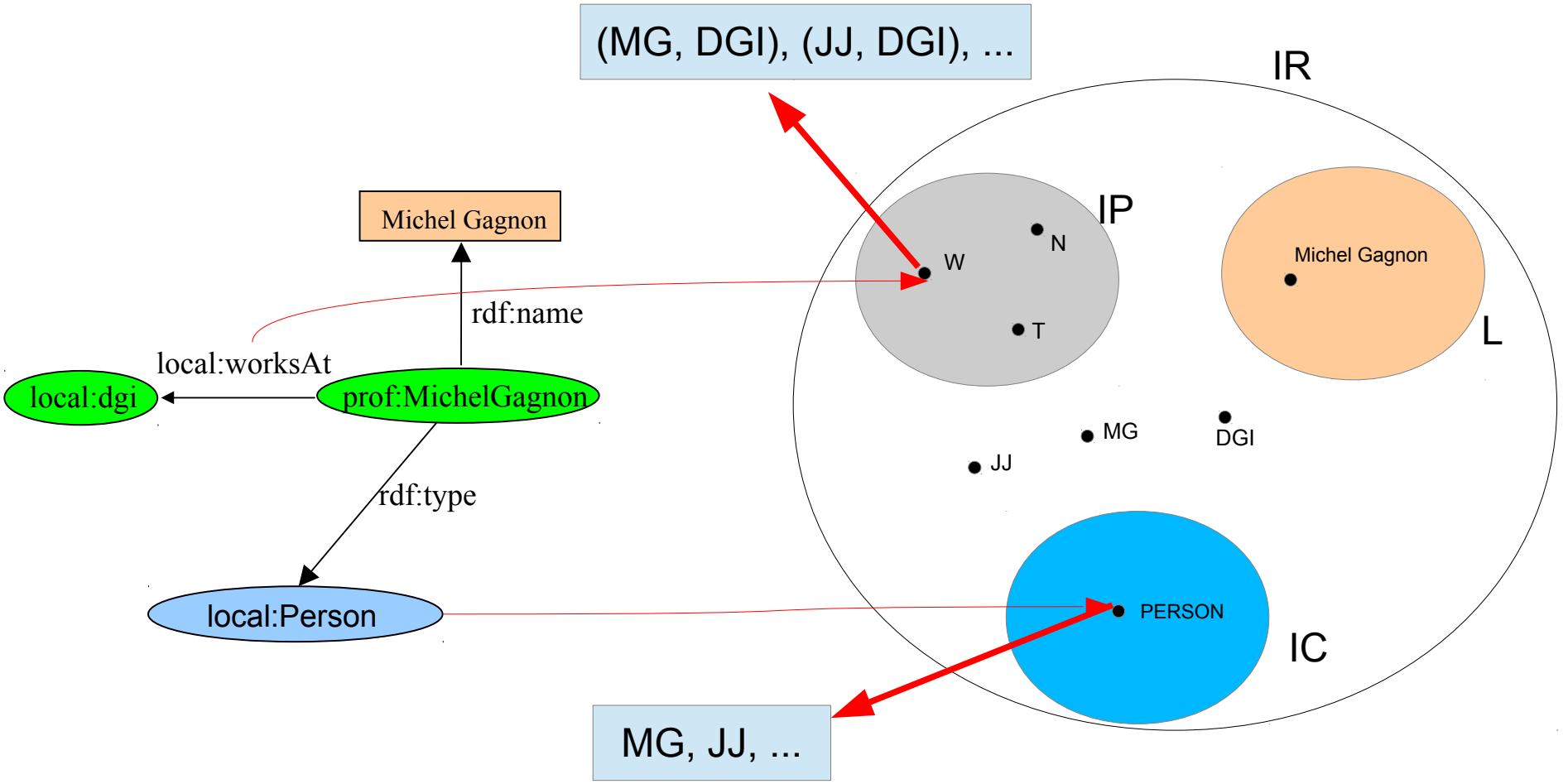
Sémantique - Exemple

À chaque URI et littéral, on associe une entité de l'univers de discours



Sémantique - Exemple

Pour chaque propriété, on a la liste des paires d'entités qui sont reliées par cette propriété



Pour chaque classe, on a la liste des entités qui en font partie

Inférence

Si on a

```
aaa p bbb .
```

On peut inférer

```
_:n1 p bbb .
```

`Si aaa` a déjà été remplacé par `_:n1` auparavant.
Sinon, il faut créer un nouveau nœud vide.

Inférence

Si on a

```
aaa p bbb .
```

On peut inférer

```
aaa p _:n1 .
```

`Si bbb` a déjà été remplacé par `_:n1` auparavant.
Sinon, il faut créer un nouveau nœud vide.

Inférence - Exemple

Soit

```
local:MichelGagnon local:worksAt local:dgi .
local:JeanStJean local:worksAt local:dgi .
```

On peut inférer

```
local:MichelGagnon local:worksAt _:n1 .
```

Inférence - Exemple

Soit

```
local:MichelGagnon local:worksAt local:dgi .
local:JeanStJean local:worksAt local:dgi .
```

On peut inférer

```
local:MichelGagnon local:worksAt _:n1 .
local:JeanStJean local:worksAt _:n1 .
```

Inférence - Exemple

Soit

```
local:MichelGagnon local:worksAt local:dgi .
local:JeanStJean local:worksAt local:dgi .
```

On peut inférer

```
local:MichelGagnon local:worksAt _:n1 .
local:JeanStJean local:worksAt _:n1 .
_:n2 local:worksAt _:n1 .
```

Inférence - Exemple

Soit

```
local:MichelGagnon local:worksAt local:dgi .
local:JeanStJean local:worksAt local:dgi .
```

On peut inférer

```
local:MichelGagnon local:worksAt _:n1 .
local:JeanStJean local:worksAt _:n1 .
_:n2 local:worksAt _:n1 .
_:n3 local:worksAt _:n1 .
```

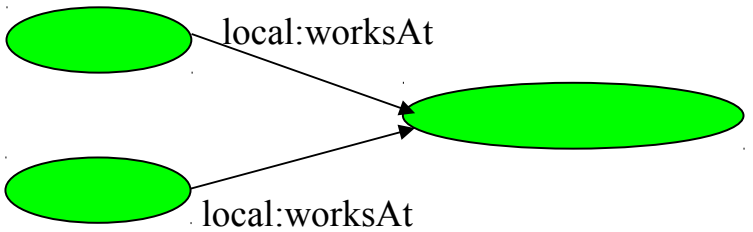
Inférence - Exemple

Ce graphe

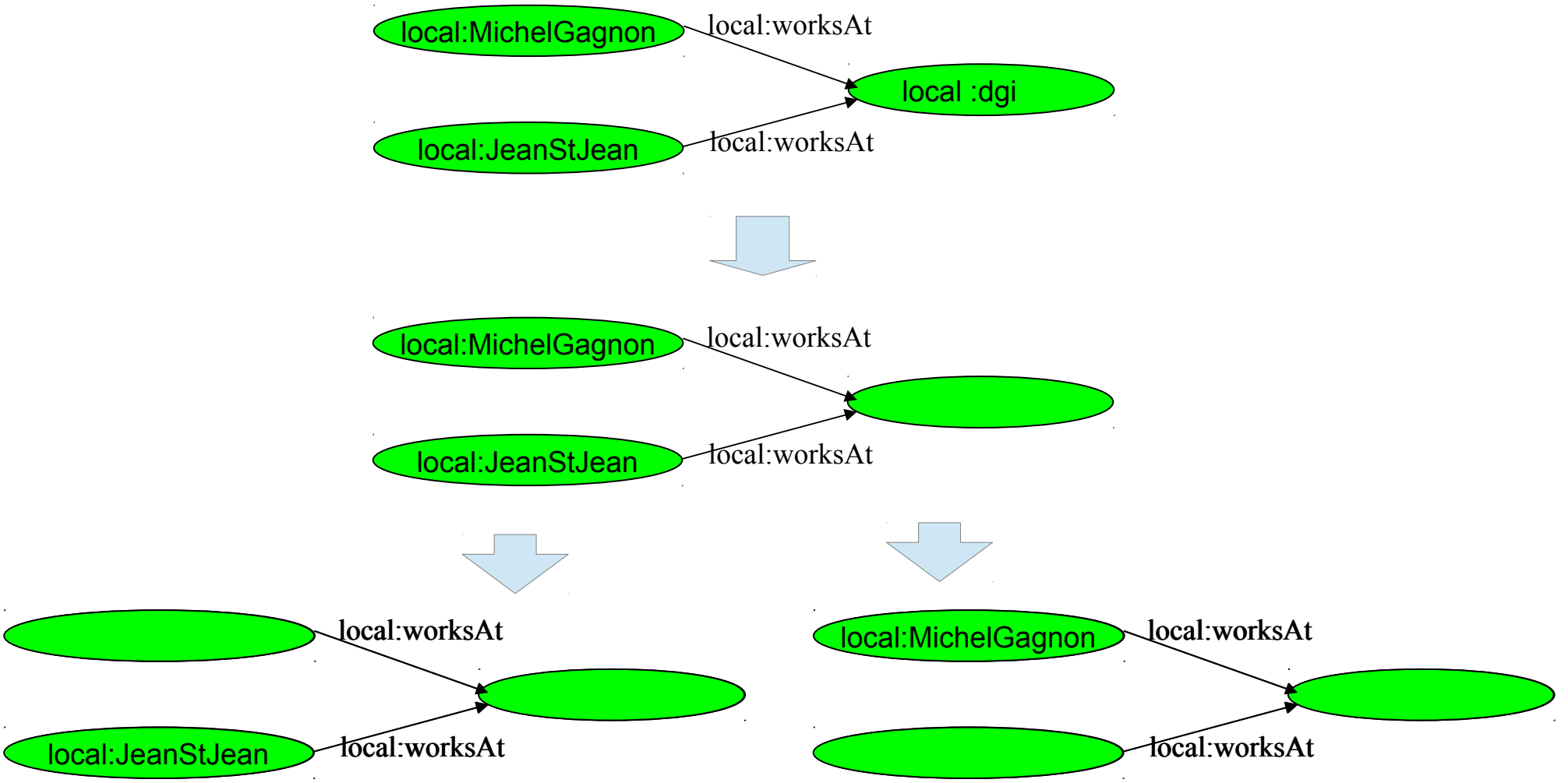
```
_:n2 local:worksAt _:n1 .
_:n3 local:worksAt _:n1 .
```

est équivalent à celui-ci

```
[] local:worksAt _:n1 .
[] local:worksAt _:n1 .
```



Inférence - Exemple



Inférence

Si on a

```
aaa p bbb .
```

On peut inférer

```
p rdf:type rdf:Property .
```

Inférence

Si on a

```
aaa p bbb .
```

On peut inférer

```
aaa rdf:type rdfs:Resource .
```

et

```
bbb rdf:type rdfs:Resource .
```

Inférence

Si on a

```
p rdfs:domain d .
aaa p bbb .
```

On peut inférer

```
aaa rdf:type d .
```

Inférence

Si on a

```
p rdfs:range d .
aaa p bbb .
```

On peut inférer

```
bbb rdf:type d .
```

Inférence - Exemple

Soit

```
local:marieAvec rdfs:domain  local:Homme .
local:marieAvec rdfs:domain  local:Femme .
local:Paul local:marieAvec local:Marie .
```

On peut inférer

```
local:Paul rdf:type local:Homme .
local:Paul rdf:type local:Femme .
```

Donc Paul est à la fois un homme et une femme

Inférence

Si on a

```
c1 rdfs:subClassOf c2 .
aaa rdf:type c1 .
```

On peut inférer

```
aaa rdf:type c2 .
```

Inférence - Exemple

Soit

```
local:Professeur rdfs:subClassOf  local:Person .
local:Michel rdf:type local:Professeur .
```

On peut inférer

```
local:Michel rdf:type local:Person .
```

Inférence

Si on a

```
c1 rdfs:subClassOf c2 .
c2 rdfs:subclassOf c3 .
```

On peut inférer

```
c1 rdf:subClassOf c3 .
```

Inférence

Si on a

```
p1 rdfs:subPropertyOf p2 .
aaa p1 bbb .
```

On peut inférer

```
aaa p2 bbb .
```

Inférence - Exemple

Soit

```
local:aime rdfs:subPropertyOf
      local:eprouveSentimentEnvers .

local:Paul rdf:aime local:Marie .
```

On peut inférer

```
local:Paul rdf:eprouveSentimentEnvers
      local:Marie .
```

Inférence

Si on a

```
p1 rdfs:subPropertyOf p2 .
p2 rdfs:subPropertyOf p3 .
```

On peut inférer

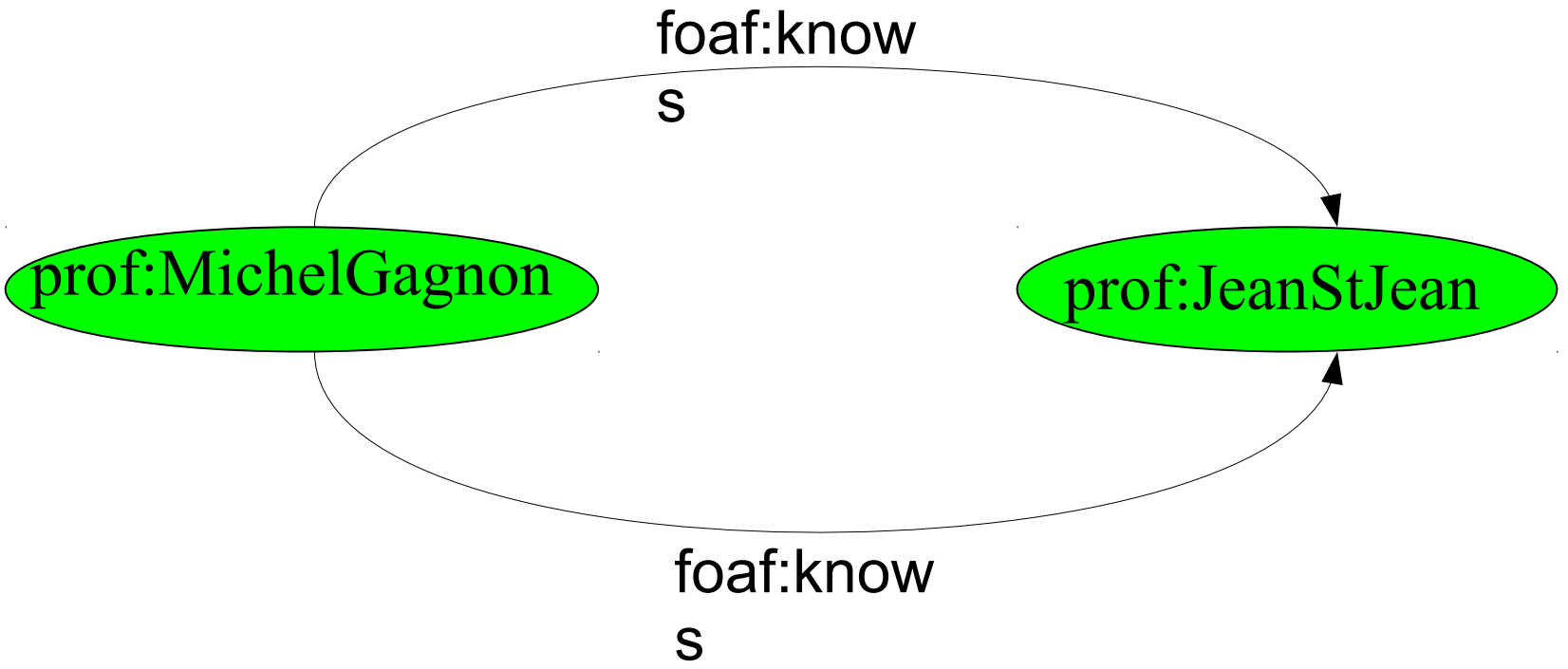
```
p1 rdf:subPropertyOf p3 .
```

Exercice – Graphe RDF valide?

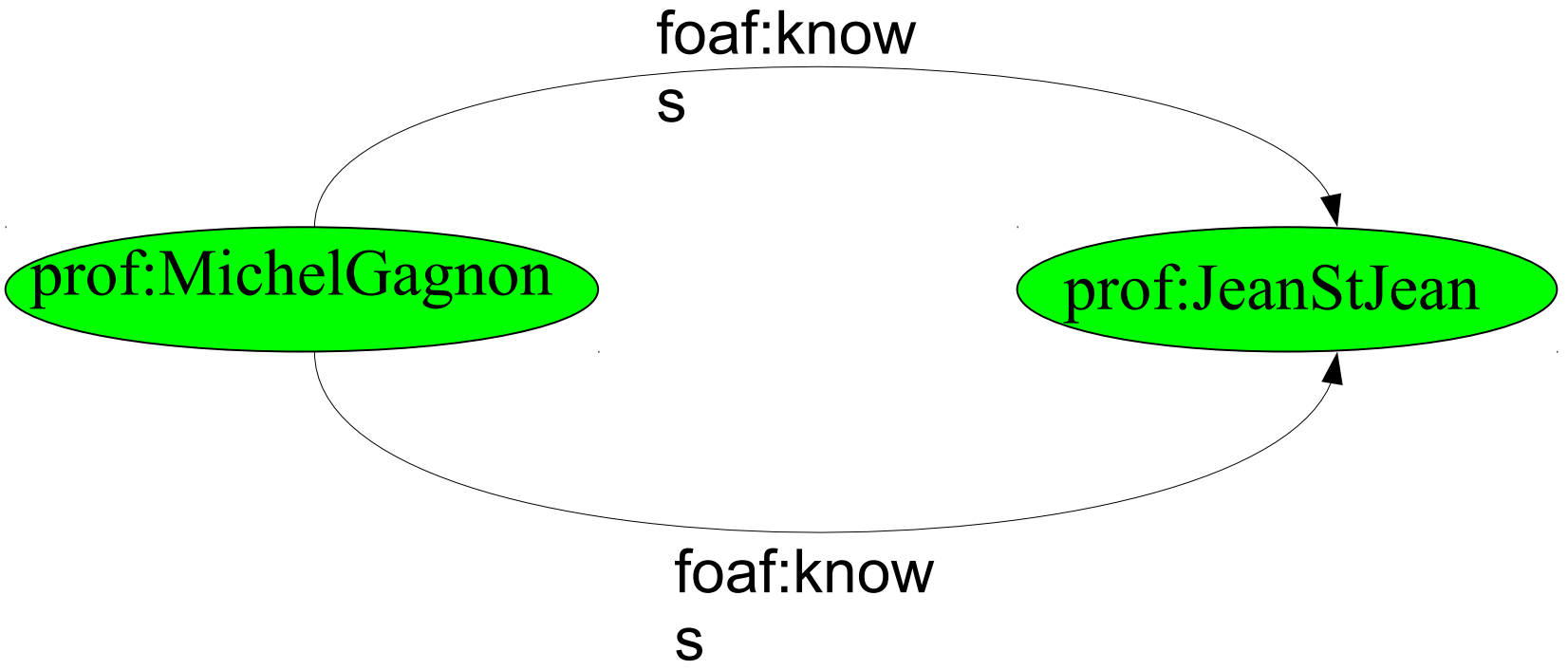
Exercice – Graphe RDF valide?

NON!

Exercice – Graphe RDF valide?

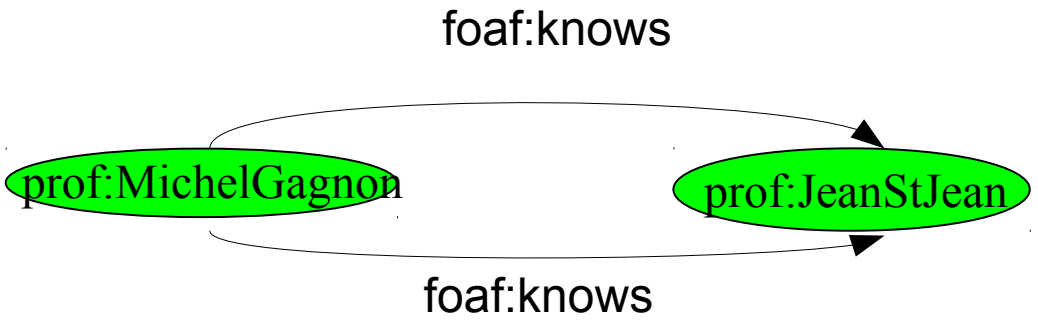


Exercice – Graphe RDF valide?

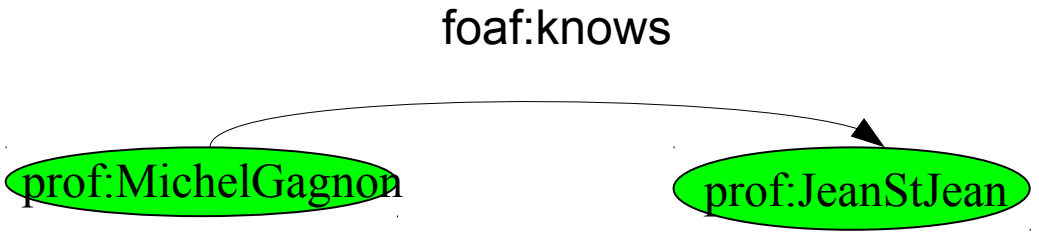


OUI!

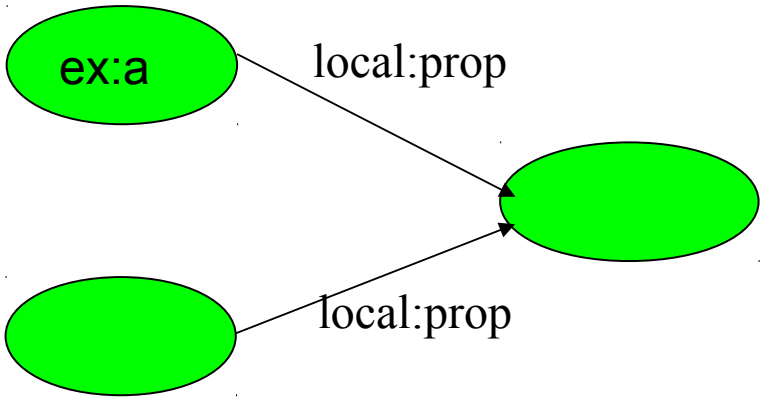
Exercice – Graphe RDF valide?

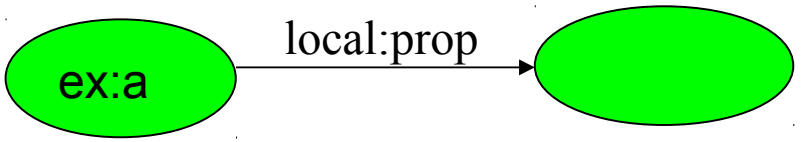


Ces deux graphes sont équivalents

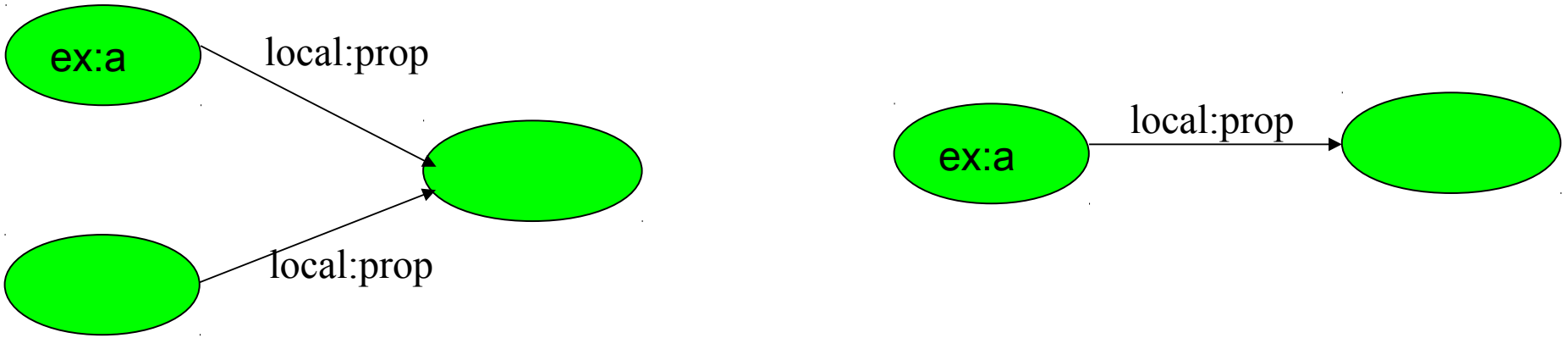


Exercice – Graphes équivalents?



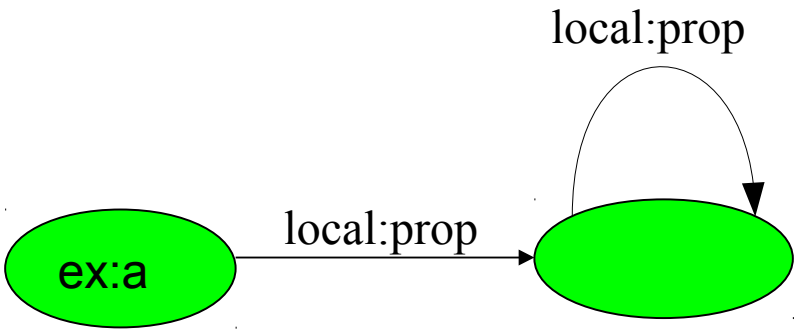


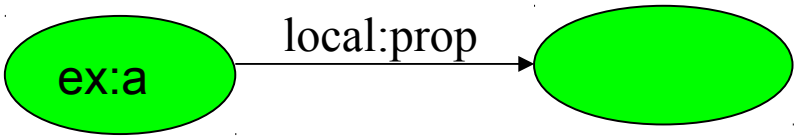
Exercice – Graphes équivalents?



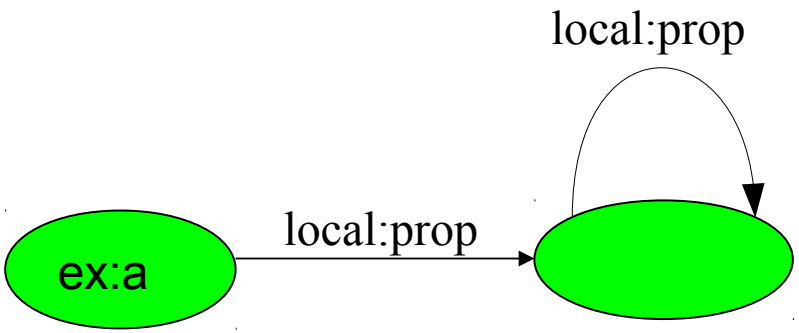
OUI!

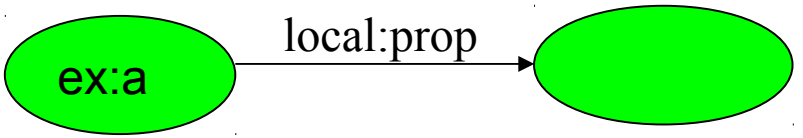
Exercice – Graphes équivalents?





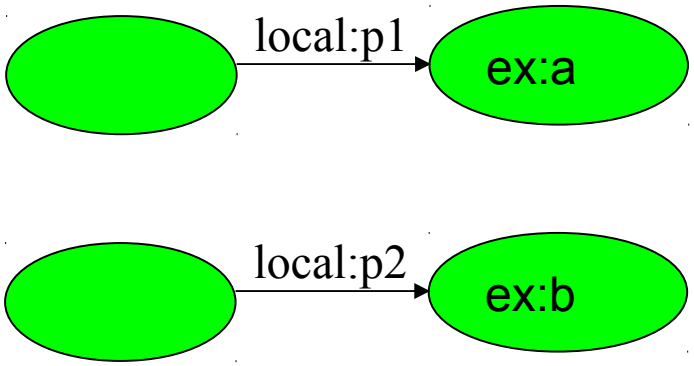
Exercice – Graphes équivalents?

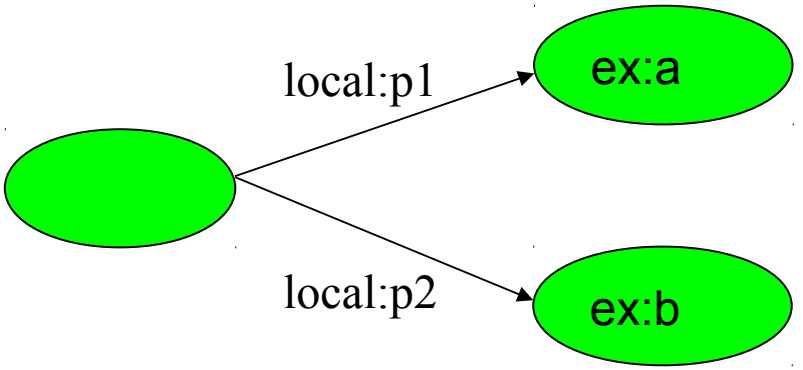




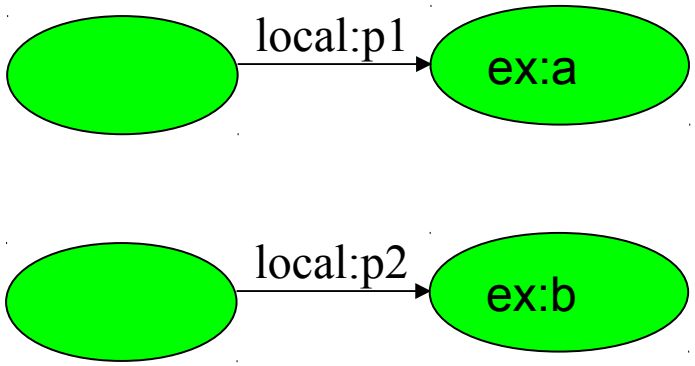
NON!

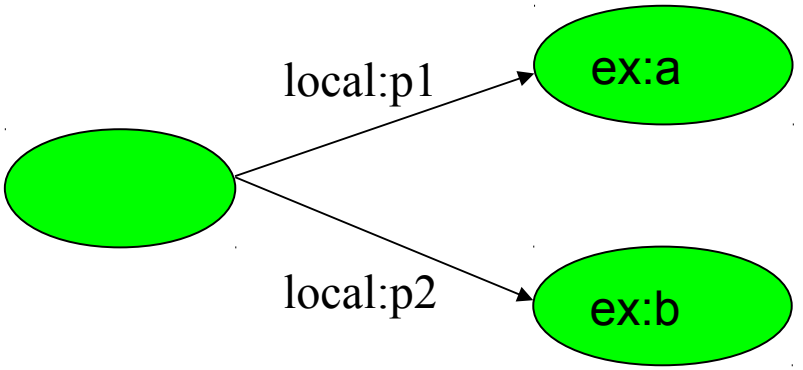
Exercice – Graphes équivalents?





Exercice – Graphes équivalents?





NON!

Montrer qu'un animal a un sentiment envers un autre animal

```
ex:detester rdfs:subPropertyOf ex:avoirSentimentEnvers .
ex:Chat rdfs:subClassOf ex:Animal .
ex:Tom rdf:type ex:Chat .
ex:Jerry rdf:type ex:Animal .
ex:Tom ex:detester ex:Jerry .
```

Monter qu'un animal a un sentiment envers un autre animal

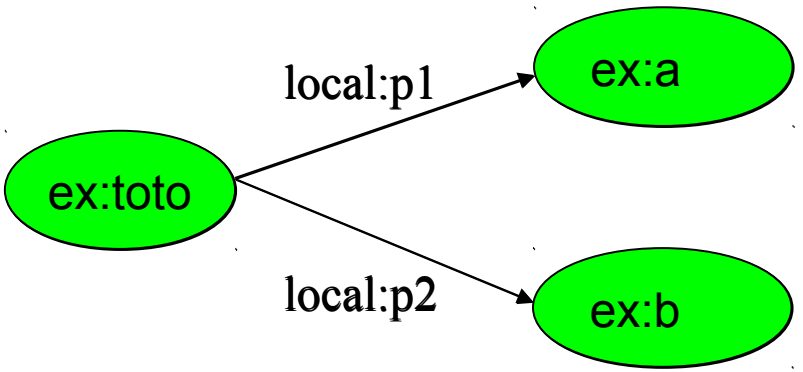
```
(1) ex:detester rdfs:subPropertyOf ex:avoirSentimentEnvers .
(2) ex:Chat rdfs:subClassOf ex:Animal .
(3) ex:Tom rdf:type ex:Chat .
(4) ex:Jerry rdf:type ex:Animal .
(5) ex:Tom ex:detester ex:Jerry .
(6) ex:Tom ex:avoirSentimentEnvers ex:Jerry .      (1,5)
(7) ex:Tom rdf:type ex:Animal .                    (2,3)
(8) _:n1 ex:avoirSentimentEnvers ex:Jerry .        (6)
(9) _:n1 ex:avoirSentimentEnvers _:n2 .            (8)
(10) _:n1 rdf:type ex:Animal .                     (7)
(11) _:n2 rdf:type ex:Animal .                     (4)
```

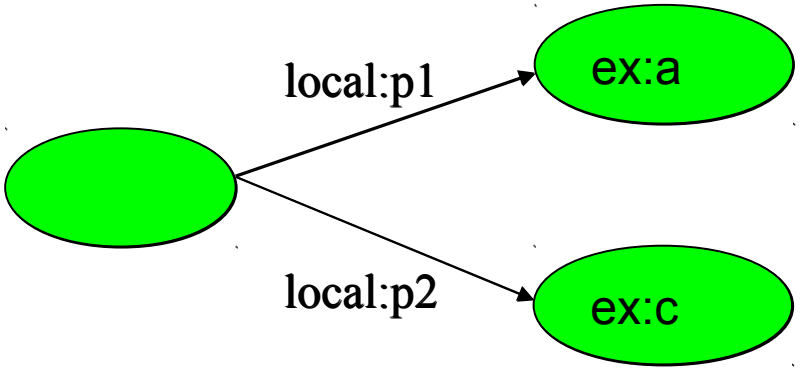
Monter qu'un animal a un sentiment envers un autre animal

```
(1) ex:detester rdfs:subPropertyOf ex:avoirSentimentEnvers .
(2) ex:Chat rdfs:subClassOf ex:Animal .
(3) ex:Tom rdf:type ex:Chat .
(4) ex:Jerry rdf:type ex:Animal .
(5) ex:Tom ex:detester ex:Jerry .
(6) ex:Tom ex:avoirSentimentEnvers ex:Jerry .      (1,5)
(7) ex:Tom rdf:type ex:Animal .                    (2,3)
(8)  :n1 ex:avoirSentimentEnvers ex:Jerry .        (6)
(9) _:n1 ex:avoirSentimentEnvers _:n2 .            (8)
(10) _:n1 rdf:type ex:Animal .                     (7)
(11) _:n2 rdf:type ex:Animal .                     (4)
```

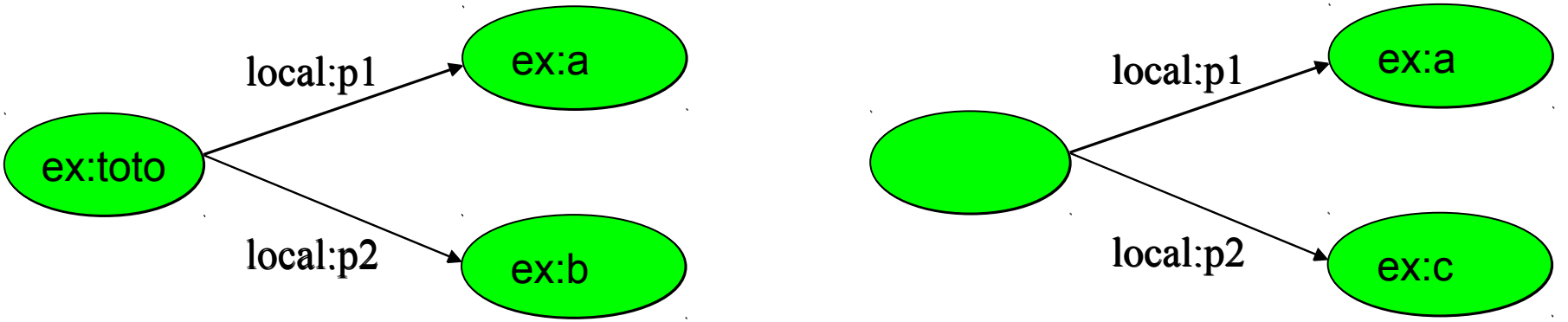
```
[ a ex:Animal ;
  ex:avoirSentimentEnvers  [  a ex:Animal ] ]  .
```

Peut-on avoir une même interprétation pour ces graphes?





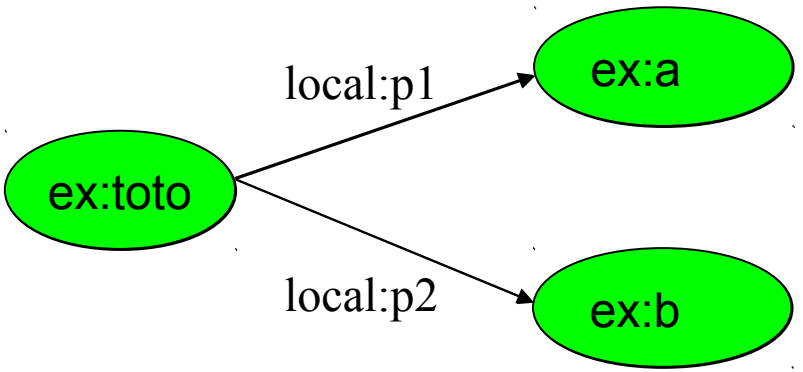
Peut-on avoir une même interprétation pour ces graphes?

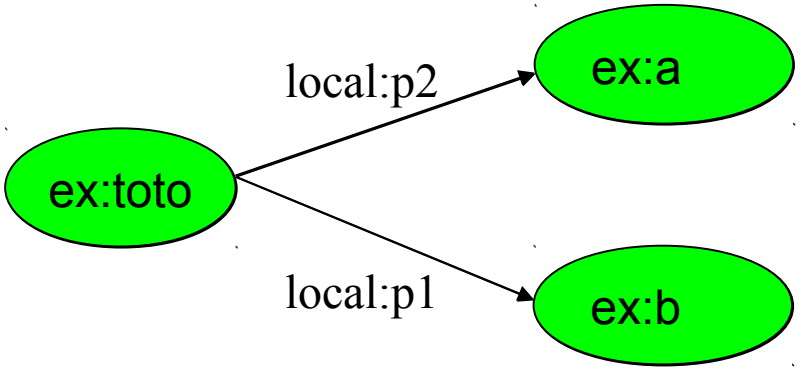


OUI!

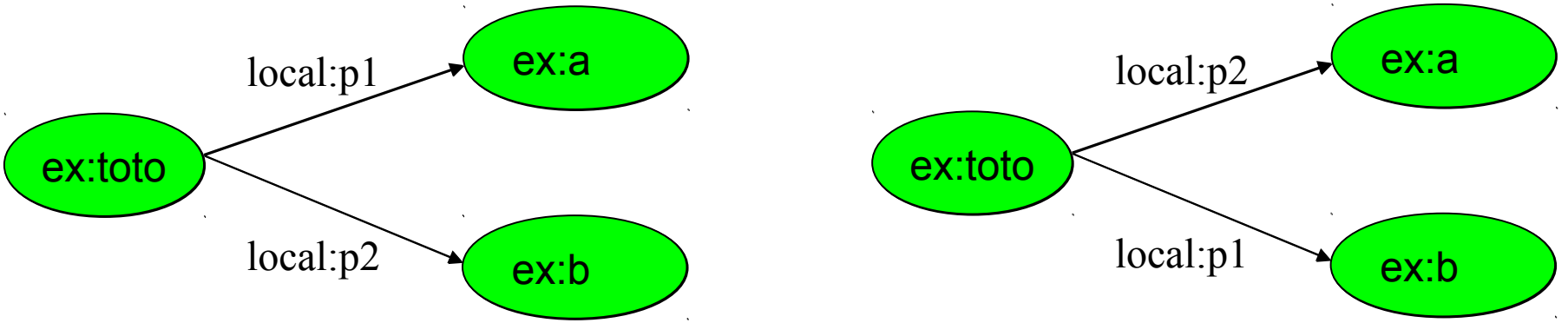
Il suffit que ex:b et ex:c désignent la même entité

Peut-on avoir une même interprétation pour ces graphes?





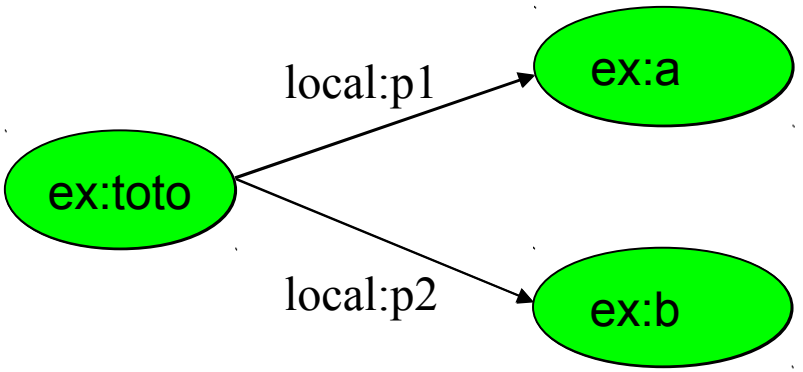
Peut-on avoir une même interprétation pour ces graphes?

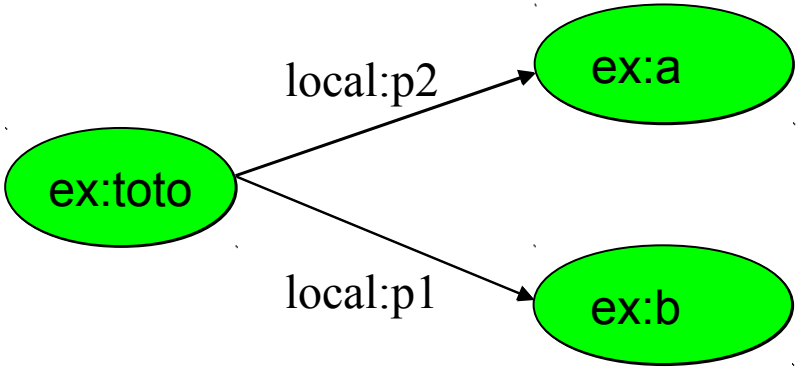


OUI!

Soient T, A et B les entités désignées par ex:toto, ex:a et ex:b, respectivement. Il suffit que les propriétés associées à local:p1 et local:p2 contiennent toutes les deux les paires (T,A) et (T,B).

Une interprétation valide pour ces 2 gaphes





sera aussi valide pour ce graphe :