Mise en oeuvre de OWL avec Protégé

Bernard ESPINASSE

Aix-Marseille Université
LIS UMR CNRS 7020

Novembre 2019

Protégé
L'ontologie "African Wildlife »
Description de classes en Protégé
Restrictions sur les propriétés en Protégé
Raisonnements sur les ontologies en Protégé

Plan

Inspiré de la présentation de J. Dibie, AgroParisTech

1. Protégé

Protégé

- **Protégé** est un système permettant la création d'ontologies très populaire développé à l'Université de Stanford
- **Protégé** est développé en Java.
- Protégé fonctionnant sur la **machine virtuelle Java,** est **multi-platefor,mes** (linux, Windows, MacOS, …)
- **Protégé** est gratuit et son code source est publié sous une licence libre (*Mozilla Public License*).
- **Protégé** peut lire et sauvegarder des ontologies dans la plupart des formats d'ontologies : RDF, RDFS, OWL, Turtle, JSON-LD, ….
- **Dernière version : 5.5.0 (14 mars 2019)**
- Lien de **téléchargement** : https://protege.stanford.edu/

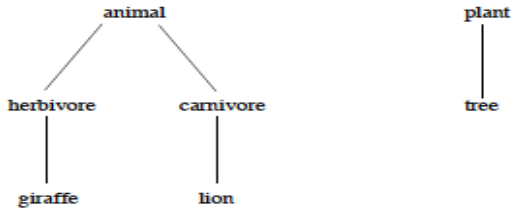
2. L'Ontologie "African Wildlife »

L'Ontologie « African Wildelife Ontology »

Source : Antoniou, G, van Harmelen, F. A Semantic Web Primer. MIT Press, 2003.

Objet :

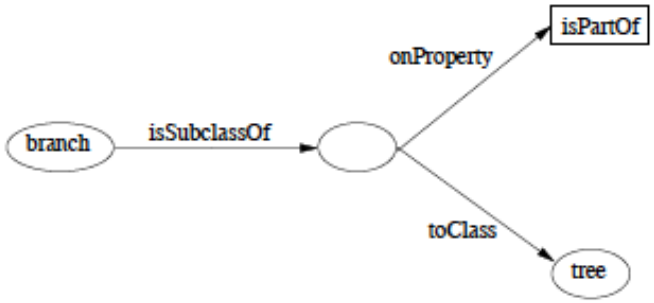
- Cette ontologie décrit la vie sauvage en Afrique avec
  - des *animaux*: *carnivores* (des *lions)* et *herbivores* (des *girafes*)
  - des *plantes* telles que des arbres composés de *branches* et de *feuilles*
  - ...
- On s'intéresse ici à un sous-ensemble de cette ontologie, centré sur les classes et sous-classes représentées par les sous-graphes suivants (class hierarchy) :



- **Remarque** : le graphe complet associé à l'ontologie est bien plus grand.

African Wildelife Ontology

- Le graphe suivant illustre le fait que les *branches* (branch) sont des *parties* (*is-part-of*) *d'arbre* (tree) :



- On aurait une représentation similaire pour indiquer que les *feuilles* (leaf) sont sont des *parties* (is-part-of) *d'arbre* (tree)

3. Description de classes en Protégé

Déclarations de classes en OWL2

OWL distingue 6 types de descriptions d'une classe :

1. Identifiant de classe (URI)
2. Enumération exhaustive de ses individus qui forment ses instances possibles
3. Intersection entre une ou plusieurs descriptions de classes
4. Union entre une ou plusieurs descriptions de classes
5. Complément d'une description de classe
6. Restriction de propriétés

1/ Par identifiant de classe (URI) en RDF/XML

```
<owl:Class rdf:ID="Animal"/>
<owl:Class rdf:ID="Herbivore">
   <rdfs:subClassOf rdf:resource="#Animal"/>
</owl:Class>
<owl:Class rdf:ID="Carnivore">
   <rdfs:subClassOf rdf:resource="#Animal"/>
</owl:Class>
...
<owl:Class rdf:ID="Plant"/>
<owl:Class rdf:ID="Tree">
   <rdfs:subClassOf rdf:resource="#Plant"/>
</owl:Class>
<owl:Class rdf:ID="Branch"/>
<owl:Class rdf:ID="Leaf"/>
```

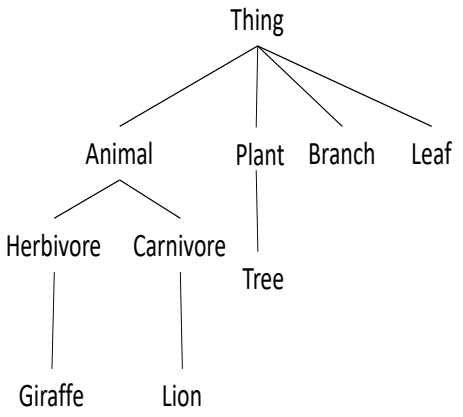
`Rdfs :subClassOf` = axiome de classe

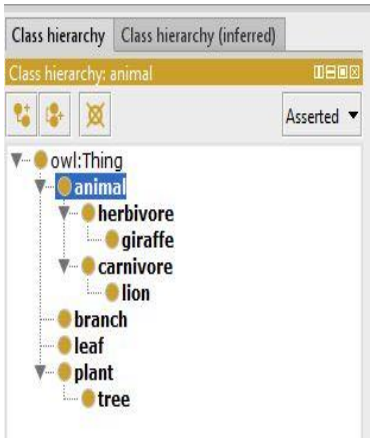
1/ Par identifiant de classe (URI) en RDF/XML

```
:Animal rdf:type owl:Class
:Herbivore rdfs:subClassOf :Animal
:Carnivore rdfs:subClassOf :Animal
:Giraffe rdfs:subClassOf :Herbivore
:Lion rdfs:subClassOf :Carnivore

:Plant rdf:type owl:Class
:Tree rdfs:subClassOf :Plant
:Branch rdf:type owl:Class
:Leaf rdf:type owl:Class
```


En Protégé





2/ Par énumération exhaustive de ses individus

- **Les instances**

```
<rdf:Description rdf:about="Ernestine_la_girafe">
   <rdf:type rdf:resource="#Giraffe"/>
</rdf:Description>
<rdf:Description rdf:about="Leon_le_lion">
   <rdf:type rdf:resource="#Lion"/>
</rdf:Description>
```

- **Plus simplement :**

```
<Giraffe rdf:about="Ernestine_la_girafe"/> <Lion
rdf:about="Leon_le_lion"/>
```

- **Turtle Syntax :**

```
:Ernestine_la_girafe rdf:type :Giraffe . :Leon_le_lion
rdf:type :Lion .
```


2/ Par énumération exhaustive d'individus : oneOf

- **Syntaxe RDF/XML :**

```
<owl:Class rdf:about="Les_girafes_copines">
<owl:equivalentClass>
<owl:Class>
<owl:oneOf rdf:parseType="Collection">
      <giraffe rdf:about="Ernestine_la_girafe"/>
      <giraffe rdf:about="Noemie"/>
      <giraffe rdf:about="Gertrude"/>
    </owl:oneOf>
  </owl:Class>
</owl:equivalentClass> </owl:Class>
```

<owl:equivalentClass> = axiome de classe

- **Syntaxe Turtle :**

```
: Les_girafes_copines owl:equivalentClass [
  rdf:type owl:Class ;
  owl:oneOf ( :Ernestine_la_girafe :Noemie :Gertrude )
].
```


Identité des instances

- Pas d'unicité des instances a priori
- Il faut déclarer explicitement si elles sont :
  - égales owl:sameAs ou
  - distinctes owl:differentFrom et owl:Alldifferent
- Dans notre exemple, nous supposons ici que les 3 instances :
  - *Ernestine_la_girafe,*
  - *Noemie*
  - *Gertrude*
- sont **distinctes**

Identité des instances

- **Syntaxe RDF/XML :**

```
<Giraffe rdf:about="Ernestine_la_girafe"/>
<Giraffe rdf:about="Noemie"/>
<Giraffe rdf:about="Gertrude"/>

<Giraffe rdf:about="Ernestine">
   <owl:sameAs rdf:resource="#Ernestine_la_girafe"/>
   <owl:differentFrom rdf:resource="#Noemie"/>
</Giraffe>

<owl:AllDifferent>
   <owl:distinctMembers rdf:parseType="Collection">
      <Giraffe rdf:about="#Ernestine_la_girafe "/>
      <Giraffe rdf:about="#Noemie"/>
      <Giraffe rdf:about="#Gertrude"/>
</owl:distinctMembers>
</owl:AllDifferent>
```

Identité des instances

- **Syntaxe Turtle :**

```
:Ernestine_la_girafe rdf:type :Giraffe .
:Noemie rdf:type :Giraffe .
:Gertrude rdf:type :Giraffe .

:Ernestine rdf:type :Giraffe .
:Ernestine owl:sameAs :Ernestine_la_girafe
:Ernestine owl:differentFrom :Noemie .

[] rdf:type owl:AllDifferent ;
   owl:distinctMembers
     ( :Ernestine_la_girafe :Noemie  :Gertrude ) .
```


3/ Par intersection de classes : intersectionOf

- **Syntaxe RDF/XML :**

```
<owl:Class rdf:about="Omnivore"> <owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Herbivore"/>
            <owl:Class rdf:about="#Carnivore"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
 </owl:Class>
```

- **Syntaxe Turtle :**

```
:Omnivore owl:equivalentClass [
   rdf:type owl:Class ;
   owl:intersectionOf ( :Herbivore :Carnivore )
].
```

- **Interprétation logique**

$\forall x \; Herbivore(x) \wedge Carnivore(x) \rightarrow Omnivore(x)$

4/ Par union de classes : unionOf

- Les **carnivores** sont définis comme **l'union** des **canidés** et des **félidés**
- **Syntaxe RDF/XML :**

```
<owl:Class rdf:about="Carnivore">
   <owl:equivalentClass>
      <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
             <owl:Class rdf:about="#Canide"/>
             <owl:Class rdf:about="#Felide"/>
          </owl:unionOf>
      </owl:Class>
   </owl:equivalentClass>
 </owl:Class>
```

- **Syntaxe Turtle :**

```
:Carnivore owl:equivalentClass [
   rdf:type owl:Class ;
   owl:unionOf ( :Canide :Felide )
].
```


5/ Par complement de classes : complementOf

- Les **canidés** sont des **carnivores** qui **ne sont pas** des **félidés**.
- **Syntaxe RDF/XML :**

```
<owl:Class rdf:about="Canide"> <owl:equivalentClass>
     <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
         <owl:Class rdf:about="#Carnivore"/>
         <owl:Class>
           <owl:complementOf rdf:resource="#Felide"/>
         </owl:Class>
       </owl:intersectionOf>
     </owl:Class>
</owl:equivalentClass> </owl:Class>
```

- **Syntaxe Turtle :**

```
:Canide owl:equivalentClass [
   rdf:type owl:Class ;
   owl:intersectionOf
       ( :Canivore [owl:complementOf :Felide ] )
].
```

Axiome de classe disjointWith

- Les **herbivores** sont **disjoints** des **carnivores**

```
<owl:Class rdf:ID="Herbivore">
   <rdfs:subClassOf rdf:ressource="#Animal"/>
   <owl:disjointWith rdf:ressource="#Carnivore"/>
</owl:Class>
```

- **Syntaxe Turtle :**

```
:Herbivore rdfs:subClassOf :Animal
:Herbivore owl:disjointwith :Carnivore
```


Axiome de classe AlldisjointWith

```
<owl:AllDisjointClasses>
  <owl:members rdf:parseType="Collection">
    <owl:Class rdf:about="Animal"/>
    <owl:Class rdf:about="Plant"/>
    <owl:Class rdf:about="Branch"/>
    <owl:Class rdf:about="Leaf"/>
  </owl:members>
</owl:AllDisjointClasses>
```

- **Syntaxe Turtle :**

```
[] rdf:type owl:AllDisjointClasses ;
   owl:members ( :Animal :Plant :Branch :Leaf ) .
```

- **Interprétation logique**

$\forall x \text{ Herbivore}(x) \rightarrow \text{Animal}(x)$

~~$\forall x \text{ Herbivore}(x) \wedge \text{non Animal}(x)$~~

- **Remarque** : `subClassOf, equivalentClass, disjointWith` et `AllDisjointWith` sont des **axiomes de classe**

6/ Par restriction de propriétés : objectproperty

- **Définissons d'abord les propriétes, objectproperty, on s'intéressera ensuite à leur restrictions**
- **Dans « African Wildlife Ontology » :**
  - les animaux *carnivores* mangent des *animaux*
  - les animaux *herbivores* mangent des *plantes* ou des *"bouts" de plantes*
- **Object property (entre 2 classes) :**
  - Object property eats
    - Domain:Animal
    - Range:Thing
- **Propriété inverse :**
  - Object property eats-by
    - Domain: Thing
    - Range: Animal

Les propriétés objets dans OWL: objectProperty

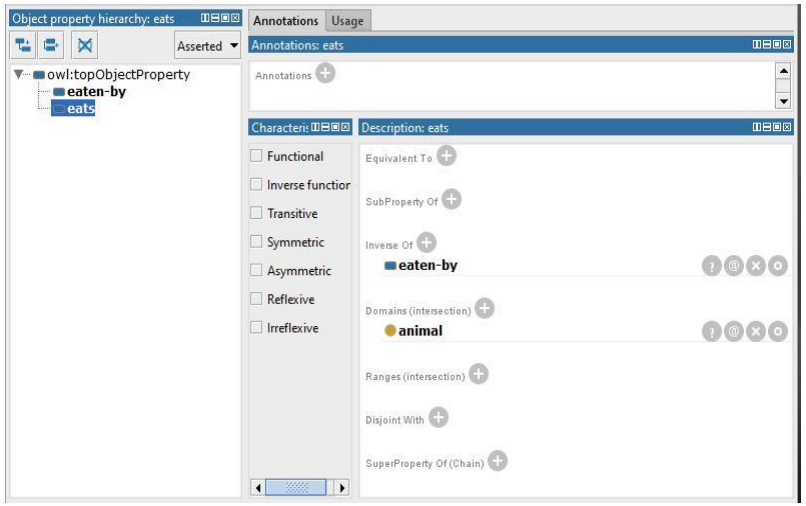
- **Syntaxe RDF/XML :**

```
<owl:ObjectProperty rdf:about="eats">
      <rdfs:domain rdf:resource="#Animal"/>
      <rdfs:range rdf:resource="Thing"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="eaten-by">
      <rdfs:range rdf:resource="#Animal"/>
      <rdfs:domain rdf:resource="Thing"/>
      <owl:inverseOf rdf:resource="#eats"/>
</owl:ObjectProperty>
```

- **Syntaxe Turtle :**

```
:eats
    rdf:type owl:ObjectProperty ;
    rdfs:domain :Animal .
:eaten-by
    rdf:type owl:ObjectProperty ;
    rdfs:range :Animal ;
    owl:inverseOf :eats .
```

Dans Protégé



Propriété OWL is-part-of

- **Dans « African Wildlife Ontology » :**
  - les animaux *carnivores* mangent des *animaux*
  - les animaux *herbivores* mangent des *plantes* ou des *"bouts" de plantes*
- **Object property (entre 2 classes) :**
  - Object property is-part-of
    - Domain:Thing
    - Range:Thing

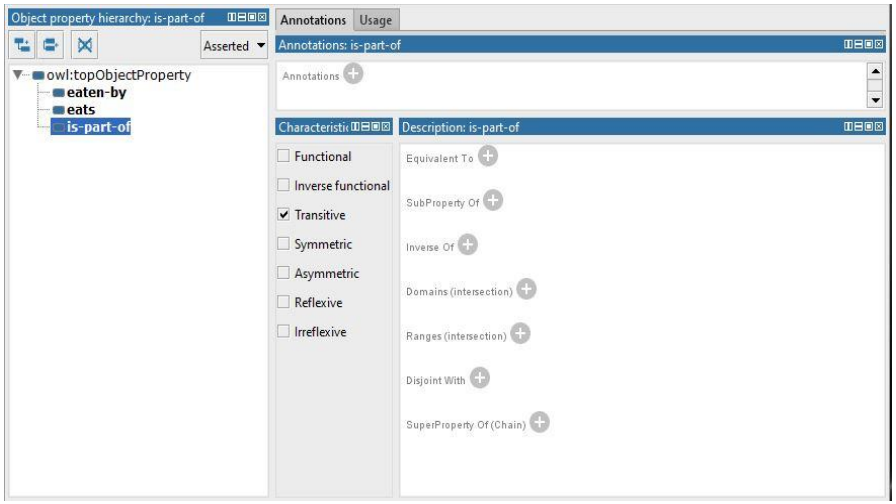
  Is-part-of = propriété **transitive**
- **Syntaxe RDF/XML :**

```
<owl:ObjectProperty rdf:about="is_part_of"/>
<owl:TransitiveProperty rdf:about="is_part_of "/>
```

- **Syntaxe Turtle :**

```
:is_part_of rdf:type owl:ObjectProperty ;
            rdf:type owl:TransitiveProperty .
```


Dans Protégé



Les propriétés de données DatatypeProperty

- **Datatype property (entre une classe et un « datatype ») :**

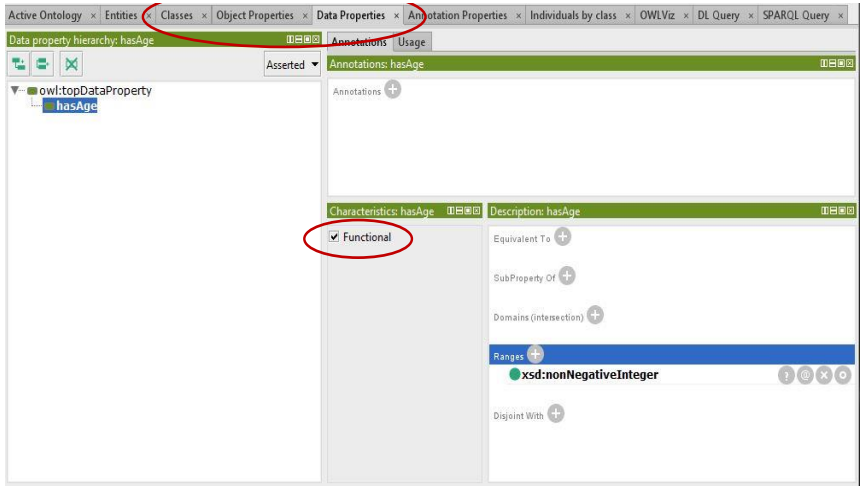
  Datatype property **hasAge**
  - Domain:**Thing**
  - Range: **nonNegativeInteger (Datatype)**
- **Syntaxe RDF/XML :**

```
<owl:DatatypeProperty rdf:about="hasAge">
   <rdfs:range rdf:datatype=
"http://www.w3.org/2001/XMLSchema#nonNegativeInteger"/>
</owl:DatatypeProperty>
```

- **Syntaxe Turtle :**

```
:hasAge rdf:type    owl:DatatypeProperty ;
        rdfs:range  xsd:NonNegativeInteger .
```

Dans Protégé



Propriété fonctionnelle : propriété qui a au plus une valeur pour chaque objet

Les propriétés de données DatatypeProperty

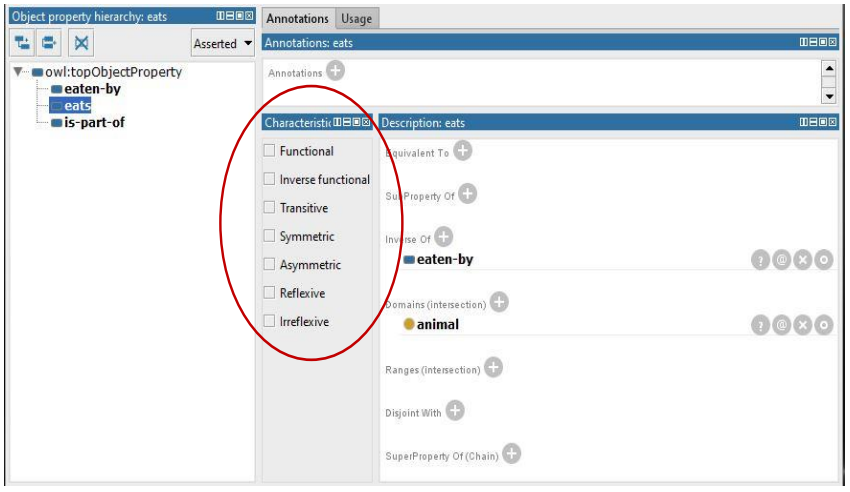
- **Syntaxe RDF/XML :**

```
<owl:DatatypeProperty rdf:about="hasAge">
   <rdfs:range rdf:resource=
   "http://www.w3.org/2001/XMLSchema#nonNegativeInteger"/>
</owl:DatatypeProperty>
<owl:FunctionalProperty rdf:about="hasAge"/>
```

- **Syntaxe Turtle :**

```
:hasAge    rdf:type     owl:DatatypeProperty ;
           rdf:type     owl:FunctionalProperty ;
           rdfs:range   xsd:NonNegativeInteger .
```


Dans Protégé: caractéristiques des propriétés



Caractéristiques des propriétés en OWL1/OWL2

- **OWL-1 :**
  - **Fonctionnelle** (au plus une valeur, e.g. ***hasAge***)

    $P(x,y) \wedge P(x,z) \rightarrow y = z$
  - **Inverse fonctionnelle** (l'inverse de la propriété est fonctionnelle, e.g. ***IsAgeOf***)

    $P(y,x) \wedge P(z,x) \rightarrow y = z$
  - **Transitive** (e.g. ***is-part-of***)

    $P(x,y) \wedge P(y,z) \rightarrow P(x,z)$
  - **Symétrique** (e.g. ***aLeMemeAge***)

    $P(x, y) \Leftrightarrow P(y, x)$
- **Spécificités OWL-2 :**
  - **Asymétrique** (OWL2) (e.g. ***estPlusGrand***)
  - **Réflexive** (OWL2) (e.g. ***aLeMemeAge***)

    $P(x, x)$
  - **Irréflexive** (OWL2) (e.g. ***estPlusGrand***)

Autres spécificités de OWL-2

- Les **chaines de propriétés**
- Les **clés**
- Définitions de **types de données plus riches**

  `owl:NegativePropertyAssertion`

  Exemple : Ernestine ne peut pas avoir 12 ans
- Possibilité de définir des **restrictions sur les cardinalités plus avancées** :

  `owl:maxQualifiedCardinality,`

  `owl:minQualifiedCardinality`

  Exemple : Léon a au moins 2 copines qui sont des copines d'Ernestines
- Amélioration des **possibilités d'annotations**

4. Restrictions sur les propriétés en Protégé

Restrictions sur les propriétés

- **Dans « African Wildlife Ontology » :**
  - les animaux *carnivores* sont des *animaux* qui mangent des *animaux*
  - Comment définir les animaux *carnivores*?
- Les *carnivores* sont des *animaux*: *subClassOf*
- Les *carnivores* sont des *animaux* qui mangent des *animaux*, **mais pas seulement**

  « *Ils mangent au moins un animal* »
- Utilisation de la restriction ***someValuesFrom***

Restrictions sur les propriétés someValuesFrom

- **Syntaxe RDF/XML :**

```
<owl:Class rdf:about="#Carnivore">
  <rdfs:subClassOf>
     <owl:Restriction>
        <owl:onProperty rdf:resource="eats"/>
        <owl:someValuesFrom rdf:resource="#Animal"/>
     </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment>
  Carnivores are exactly those animals that eat also animals.
  </rdfs:comment>
</owl:Class>
```

- **Remarque** : les restrictions sont toujours définies à partir d'une classe sur une propriété

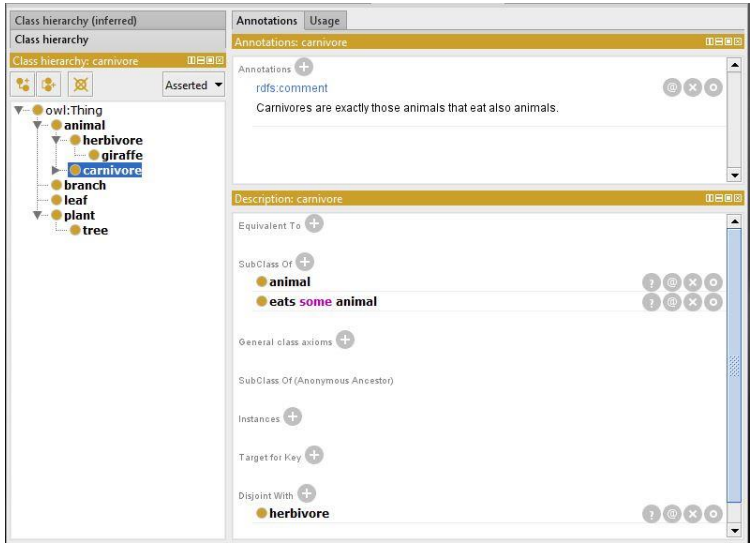
Restrictions sur les propriétés **someValuesFrom**

- **Syntaxe RDF/XML :**

```
:Carnivore owl:subClassOf [
   rdf:type owl:Restriction ;
   owl:onProperty :eats ; owl:someValuesFrom :Animal
];

<rdfs:comment>
  Carnivores are exactly those animals that eat also animals.
```


Dans Protégé



Restrictions sur les propriétés **allValuesFrom**

- **Dans « African Wildlife Ontology » :**
  - Les *herbivores* sont des *animaux*: *subClassOf*
  - Les *herbivores* sont des *animaux* qui ne mangent que :
    - des *plantes* ou
    - des *"bouts" de plantes*
  - **Comment définir les animaux herbivores?**

    ⇨ Utilisation de la restriction ***allValuesFrom***

Restrictions sur les propriétés **allValuesFrom**

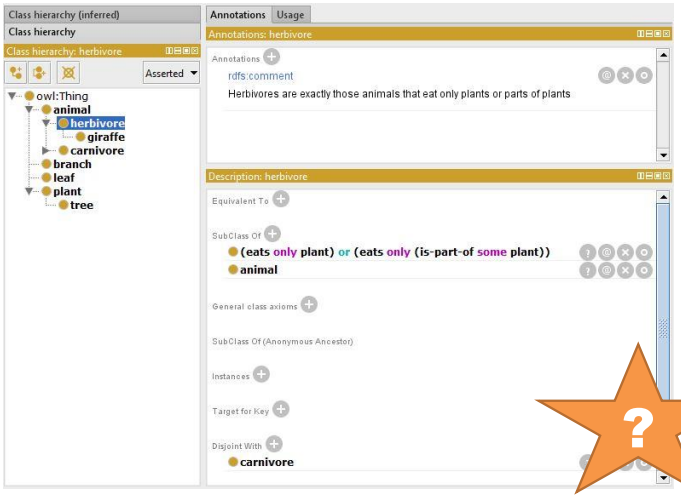
- **Syntaxe RDF/XML :**

```
<owl:Class rdf:about="#Herbivore">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="eats"/>
      <owl:allValuesFrom rdf:resource="#Plant"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment>
   Herbivores are exactly those animals that eat
only plants or parts of plants.
  </rdfs:comment>
</owl:Class>
```

- **Syntaxe Turtle :**

```
:Herbivore owl:subClassOf [
           rdf:type           owl:Restriction ;
           owl:onProperty     :eats ;
           owl:allValuesFrom  :Plant
];
```

Dans Protégé



Ensembles de restrictions sur une classe

- Classe Herbivore :

```
<owl:Class rdf:about="#Herbivore">
  <owl:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="eats"/>
          <owl:allValuesFrom rdf:resource="#Plant"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="eats"/>
          <owl:allValuesFrom>
            <owl:Restriction>
              <owl:onProperty rdf:resource="is-part-of"/>
              <owl:someValuesFrom rdf:resource="#Plant"/>
            </owl:Restriction>
          </owl:allValuesFrom>
        </owl:Restriction>
      </owl:unionOf>
    </owl:Class>
  </owl:subClassOf>
</owl:Class>
```


Restrictions sur les propiétés : HasValue

- Les *herbivores* « *basilicéens* » sont des herbivores qui ne mangent que du *basilic*, le *basilic* étant une instance de la classe *plant*
- **Utilisation de la restriction *hasValue***

```
<Plant rdf:about="Basilic"/>
<owl:Class rdf:about ="Herbivore_Basiliceen">
  <owl:equivalentClass>
     <owl:Restriction>
       <owl:onProperty rdf:resource="eats"/>
       <owl:hasValue rdf:resource="Basilic"/>
     </owl:Restriction>
   </owl:equivalentClass>
 </owl:Class>
```

- **Syntaxe Turtle :**

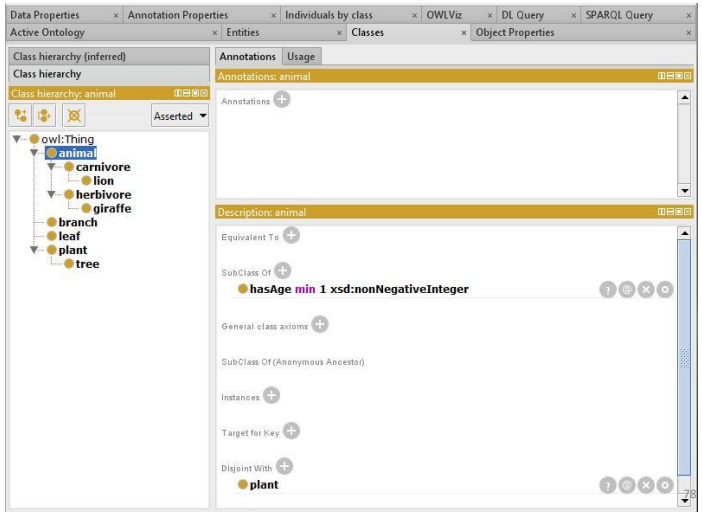
```
:Basilic rdf:type :Plant
:Herbivore_Basiliceen owl:equivalentClass [
       rdf:type          owl:Restriction ;
       owl:onProperty    :eats ;
       owl:hasValue      :Basilic
```


Restriction sur les cardinalités

- Dans l'African Wildlife Ontology : *tout animal a un âge et un seul*
- Utilisations des ***contraintes de cardinalité*** :
  - *minCardinality (au moins)*
  - *maxCardinality (au plus)*
  - *cardinality (exactement)*
- Syntaxe RDF/XML :

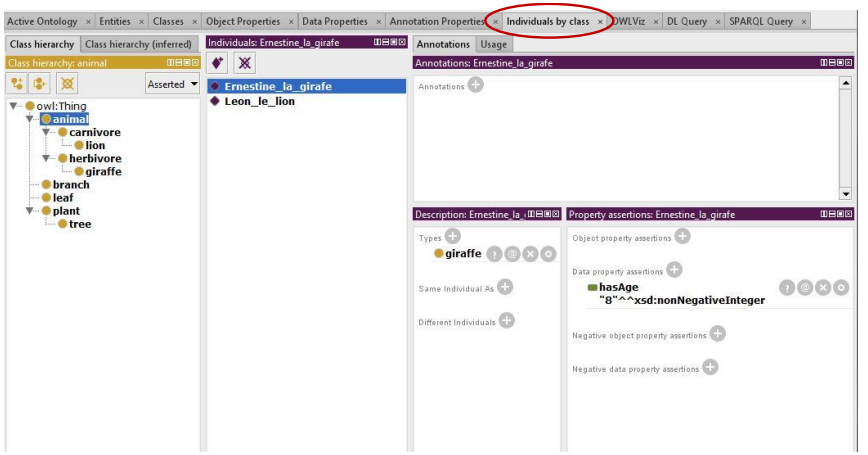
```
<owl:Class rdf:about="#Animal">
 <rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="hasAge"/>
     <owl:cardinality
       rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1
     </owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

Dans Protégé



Les instances dans Protégé

- Saisie des instances dans Protégé :



Spécificités de OWL2

- **Saisie des instances dans Protégé :**
- **Les chaines de propriétés**
- **Les clés**
- **Définitions de types de données plus riches :**

  `owl:NegativePropertyAssertion`

  Exemple : Ernestine ne peut pas avoir 12 ans

  - Possibilité de définir des restrictions sur les cardinalités plus avancées

    `owl:maxQualifiedCardinality`
    `owl:minQualifiedCardinality`

    Exemple : Léon a au moins 2 copines qui sont des copines d'Ernestines

- **Amélioration des possibilités d'annotations**

Spécificités de OWL2: **chaînes de propriétés**

```
<rdf:Description rdf:about="hasGrandparent">
    <owl:propertyChainAxiom  rdf:parseType="Collection">
      <owl:ObjectProperty rdf:about="hasParent"/>
      <owl:ObjectProperty rdf:about="hasParent"/>
   </owl:propertyChainAxiom>
 </rdf:Description>
```

- **Syntaxe Turtle :**

```
:hasGrandparent  owl:propertyChainAxiom
```

Spécificités de OWL2: **clés**

```
<owl:Class rdf:about="Person">
    <owl:hasKey rdf:parseType="Collection">
        <owl:ObjectProperty rdf:about="hasSSN"/>
    </owl:hasKey>
</owl:Class>
```

- **Syntaxe Turtle :**

```
:Person owl:hasKey ( :hasSSN ) .
```

Spécificités de OWL2: annotations

```
<owl:Class rdf:about="Man">
   <rdfs:subClassOf rdf:resource="Person"/>
</owl:Class>
<owl:Axiom>
   <owl:annotatedSource rdf:resource="Man"/>
   <owl:annotatedProperty
      rdf:resource="&rdfs;subClassOf"/>
<owl:annotatedTarget rdf:resource="Person"/>
<rdfs:comment>States that every man is a person.
</rdfs:comment>
</owl:Axiom>
```

- **Syntaxe Turtle :**

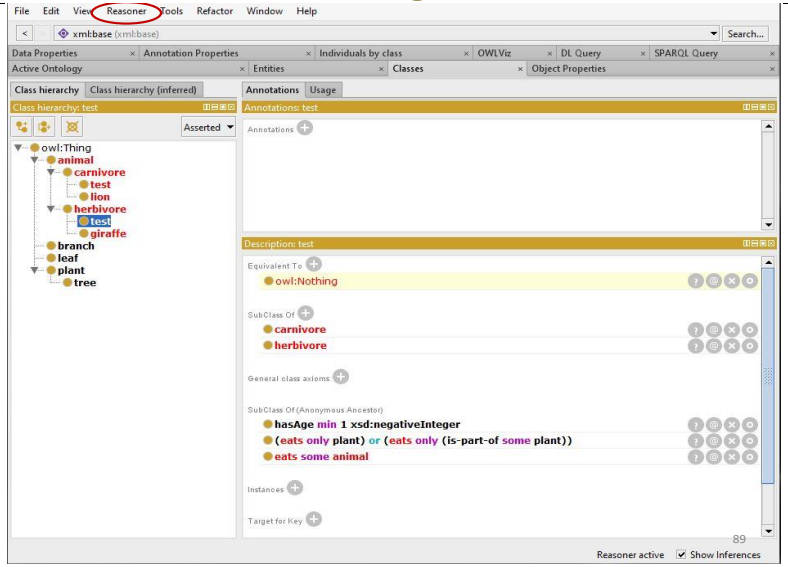
```
:Man   rdfs:subClassOf :Person .
 []    rdf:type        owl:Axiom ;
       owl:annotatedSource     :Man ;
       owl:annotatedProperty rdfs:subClassOf ;
       owl:annotatedTarget     :Person ;
       rdfs:comment "States that every man is a
           person."^^xsd:string .
```


5. Raisonnements sur les ontologies

Raisonnements sur les ontologies

- **Pourquoi raisonner sur les ontologies :**
  - Pour **vérifier la cohérence** d'une ontologie
  - Pour **inférer de nouvelles connaissances**, on peut utiliser un raisonneur
  - Exemple :
    - supposons qu'on crée une nouvelle classe ***test***, sous-classe de *Carnivore* et *Herbivore* qui ont été déclarées *disjointes*.
    - Si on lance le « raisonneur » de Protégé, on obtient...

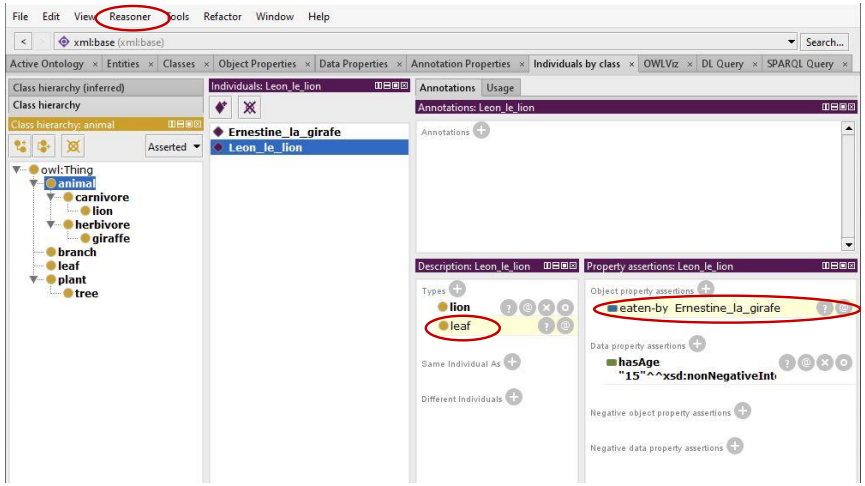
Raisonnements dans Protégé



Raisonnements sur les ontologies

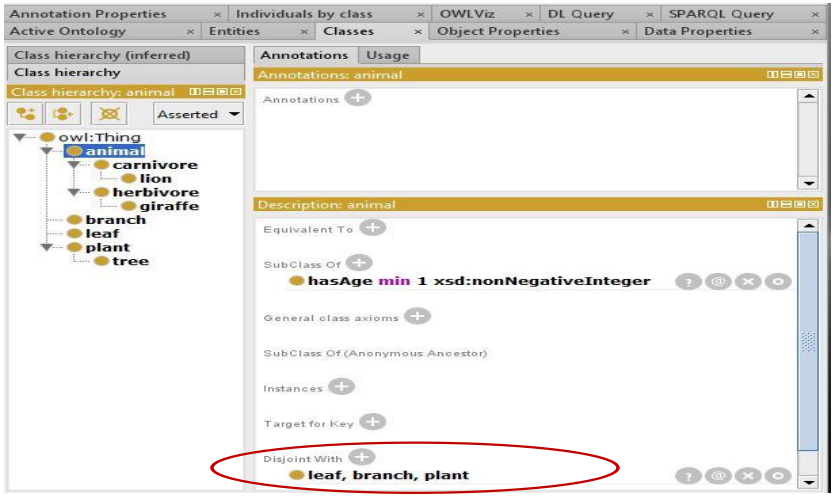
- **Supposons** qu'*Ernestine_la_girafe* mange *Leon_le_lion*, on obtient...
- **Nouvelles connaissances inférées** :
  - Ernestine_la_giraffe is-a Giraffe
  - Ernestine_la_giraffe is-a Herbivore
  - Ernestine_la_giraffe (eats only Plant) or (eats only (is-part-of some Plant))
  - **Leon_le_lion is-a Leaf.** **!**
  - **Leon_le lion eaten-by Ernestine_la_giraffe.** **!**
- Possible car les classes *Lion* et *Leaf* ne sont *pas disjointes* !

Avec Protégé



Raisonnement dans Protégé

- Ajoutons que les classes *Leaf* et *Animal* sont disjointes, on obtient :



Raisonnement dans Protégé

- Exemple de 2 justifications sur 4 trouvées par le raisonneur de Protégé 5.0 :