

Web sémantique et Ontologie

Le langage OWL



U.F.R. d'informatique

Juliette Dibie

What is the semantic Web ?

- The Semantic Web is a **Web of Data**. The collection of Semantic Web technologies (RDF, OWL, SKOS, SPARQL, etc.) provides an environment where application can query that data, draw inferences using vocabularies, etc.
- However, to make the Web of Data a reality, it is important to have the huge amount of **data** on the Web **available** in a standard format, reachable and manageable by Semantic Web tools.
- Furthermore, not only does the Semantic Web need access to data, but **relationships among data** should be made **available**, too, to create a Web of Data (as opposed to a sheer collection of datasets). This collection of interrelated datasets on the Web can also be referred to as **Linked Data**.

The Linked Data

- Il existe de nombreux standards et ontologies de référence !
 - Dublin Core metadata, FOAF ontology, PROV ontology, Time ontology, SKOS, ...
 - Foundation ontologies : BFO, DOLCE, SUMO...
- Besoin de respecter des standards !

The 5 stars Linked Data



Publish data on the Web in any format, with an open license (e.g. PDF file)



Use structured data formats (e.g. Excel file)



Use non-proprietary formats (e.g. CSV file instead of Excel)



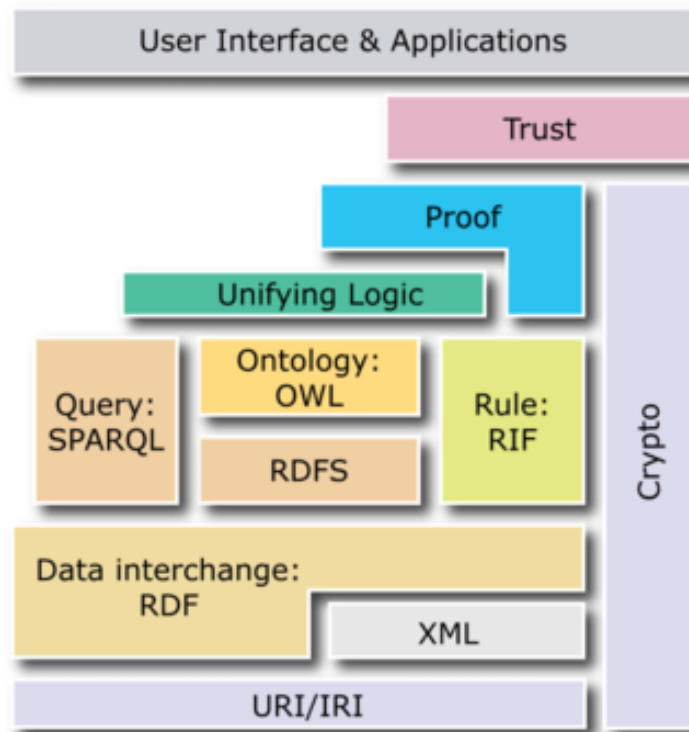
Use open standards from W3C to represent data (e.g. RDF and OWL)



Link your data to other data sets on the Web for providing context

The Web Ontology Language (OWL)

- OWL is part of the W3C's Semantic Web technology stack, which includes [RDF](#), [RDFS](#), [SPARQL](#), etc.



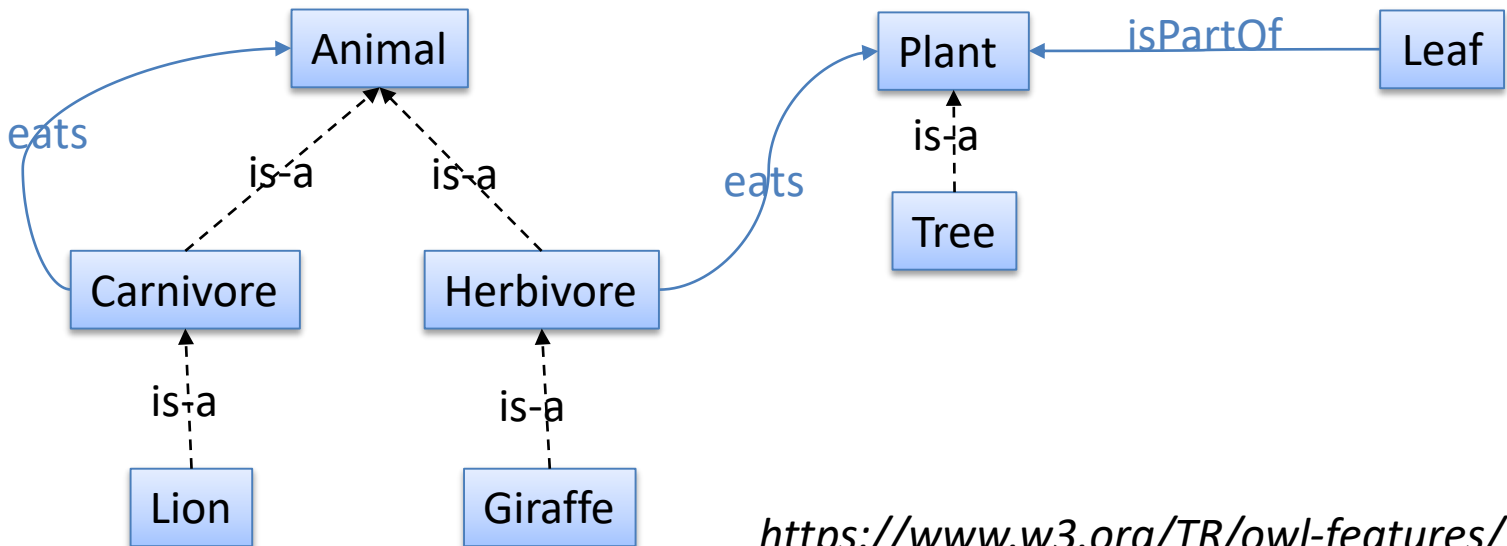
The Web Ontology Language (OWL)

- The W3C Web Ontology Language (OWL) is a Semantic Web language designed **to represent rich and complex knowledge** about things, groups of things, and relations between things.
- OWL is a **computational logic-based language** such that knowledge expressed in OWL can be exploited by computer programs.
- OWL documents are called **ontologies**. Ontologies are formalized vocabularies, often covering a specific domain and shared by a community of users.

Un exemple

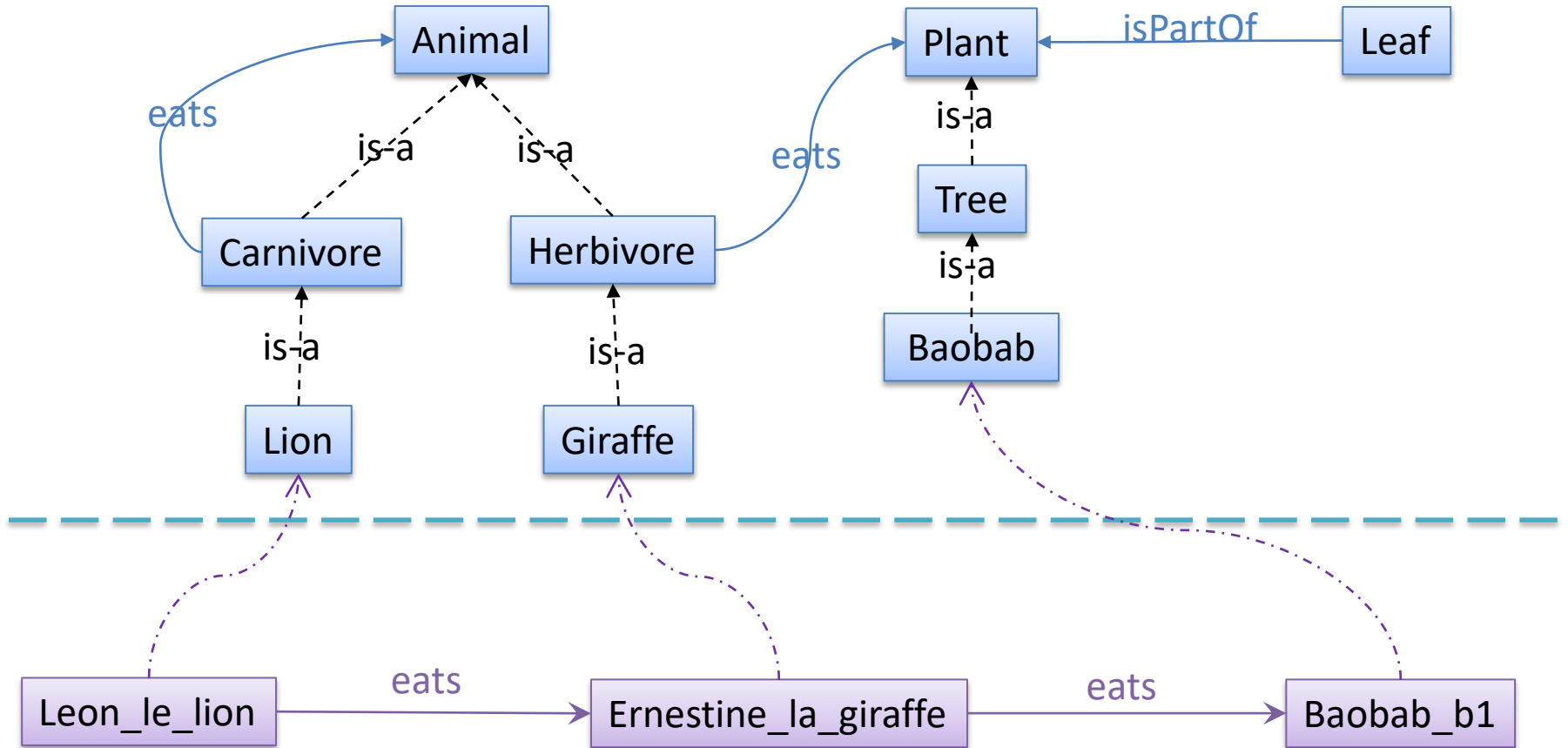
The African Wildlife Ontology

- Antoniou, G, van Harmelen, F. [A Semantic Web Primer](#). MIT Press, 2003.
- Cette ontologie décrit la vie sauvage en Afrique avec
 - des animaux carnivores tels que des lions et herbivores tels que des girafes
 - des plantes telles que des arbres composés de branches et de feuilles...



Un exemple

The African Wildlife Ontology



Sommaire

- **Les classes OWL**
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Définition d'une classe

En-tête - syntaxe RDF/XML

```
<?xml version="1.0"?>
```

En-tête: un document OWL est un document RDF

```
<rdf:RDF
```

```
  xmlns="http://www.owl-ontologies.com/AfricanWildlifeOntology1.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.owl-
```

```
  ontologies.com/AfricanWildlifeOntology1.owl#">
```

```
<!-- Definition de la classe http://www.owl-
```

```
  ontologies.com/AfricanWildlifeOntology1.owl#Animal -->
```

```
<owl:Class rdf:ID="Animal"/>
```

```
  ...
```

*Description d'une classe
par son identifiant*

```
</rdf:RDF>
```

Définition d'une classe

En-tête - syntaxe Turtle

En-tête avec la syntaxe Turtle

```
@prefix : <xml:base#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix base: <http://www.owl-
  ontologies.com/AfricanWildlifeOntology0.owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<xml:base> rdf:type owl:Ontology .

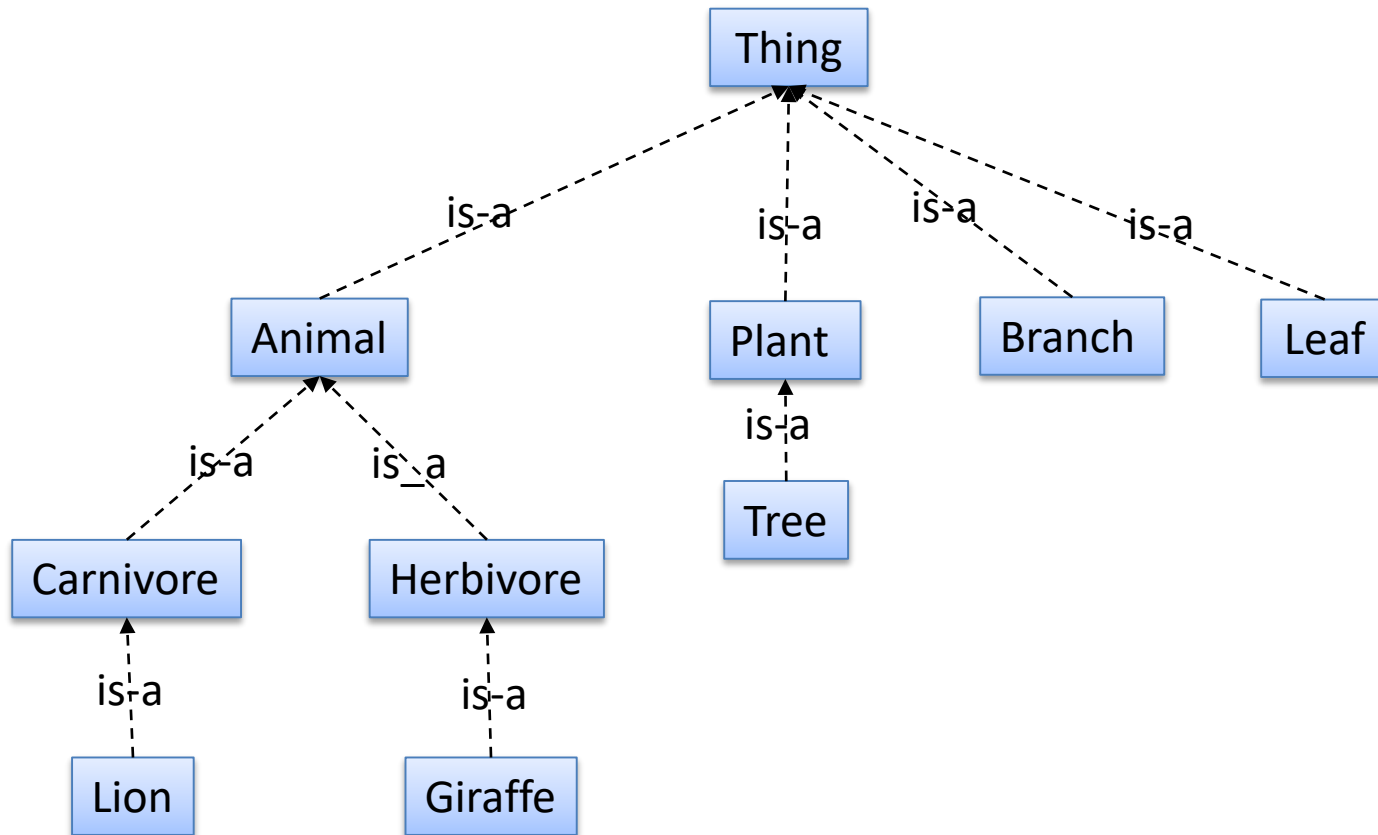
# Definition de la classe xml:base#Animal
:Animal rdf:type owl:Class .

...
```

Sommaire

- Les classes OWL
 - **Définition par subsomption**
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Les classes OWL



Définition d'une classe par subsomption - syntaxe 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"/>
```

Axiome de classe

Définition d'une classe par subsomption - syntaxe Turtle

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

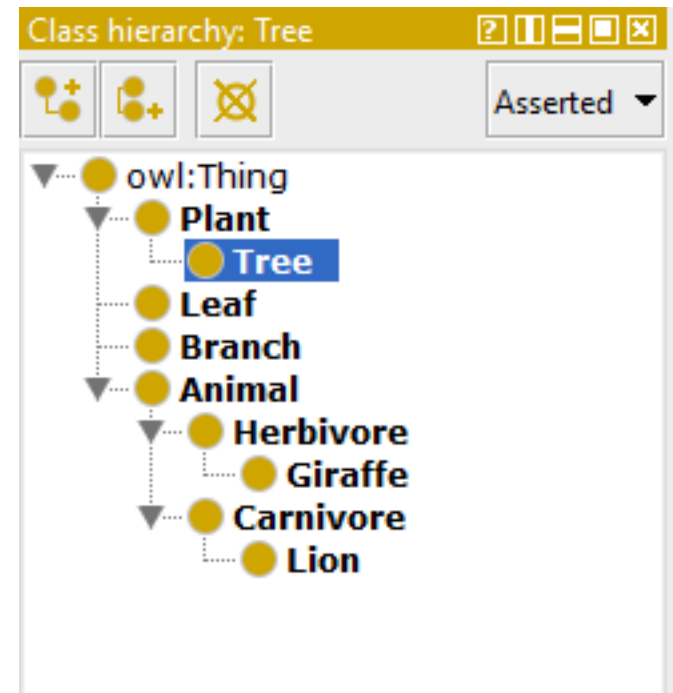
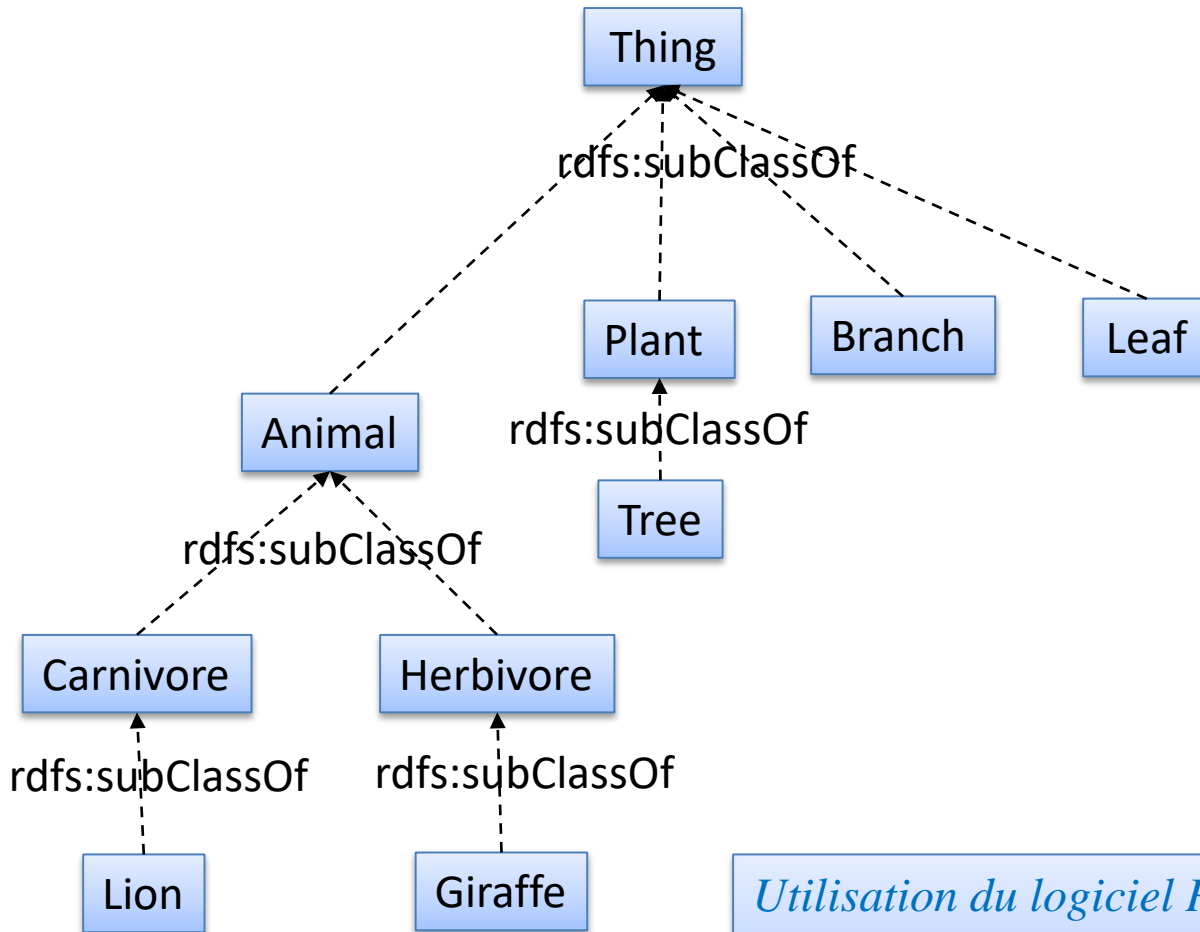
Axiome de classe

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

Interprétation logique

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

Définition d'une classe par subsomption – logiciel Protégé



Utilisation du logiciel Protégé, qui est un éditeur d'ontologie libre (<http://protege.stanford.edu/>), reposant sur le langage Java

Sommaire

- Les classes OWL
 - Définition par subsomption
 - **Définition par disjonction**
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Définition d'une classe par disjonction

- Les herbivores sont disjoints des carnivores

RDF/XML Syntax

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

Axiome de classe

Turtle Syntax

```
:Herbivore    rdfs:subClassOf :Animal ;  
              owl:disjointWith :Carnivore .
```

Interprétation logique

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

Disjonction entre plusieurs classes

RDF/XML Syntax

```
<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>
```

Axiome de classe

Turtle Syntax

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


Disjonction entre plusieurs classes

Turtle Syntax

```
:Herbivore    rdfs:subClassOf :Animal .  
:Carnivore    rdfs:subClassOf :Animal .  
[] rdf:type    owl:AllDisjointClasses ;  
    owl:members ( :Animal :Herbivore :Carnivore ) .
```

Qu'en pensez-vous ?



Justifiez votre réponse par une interprétation logique

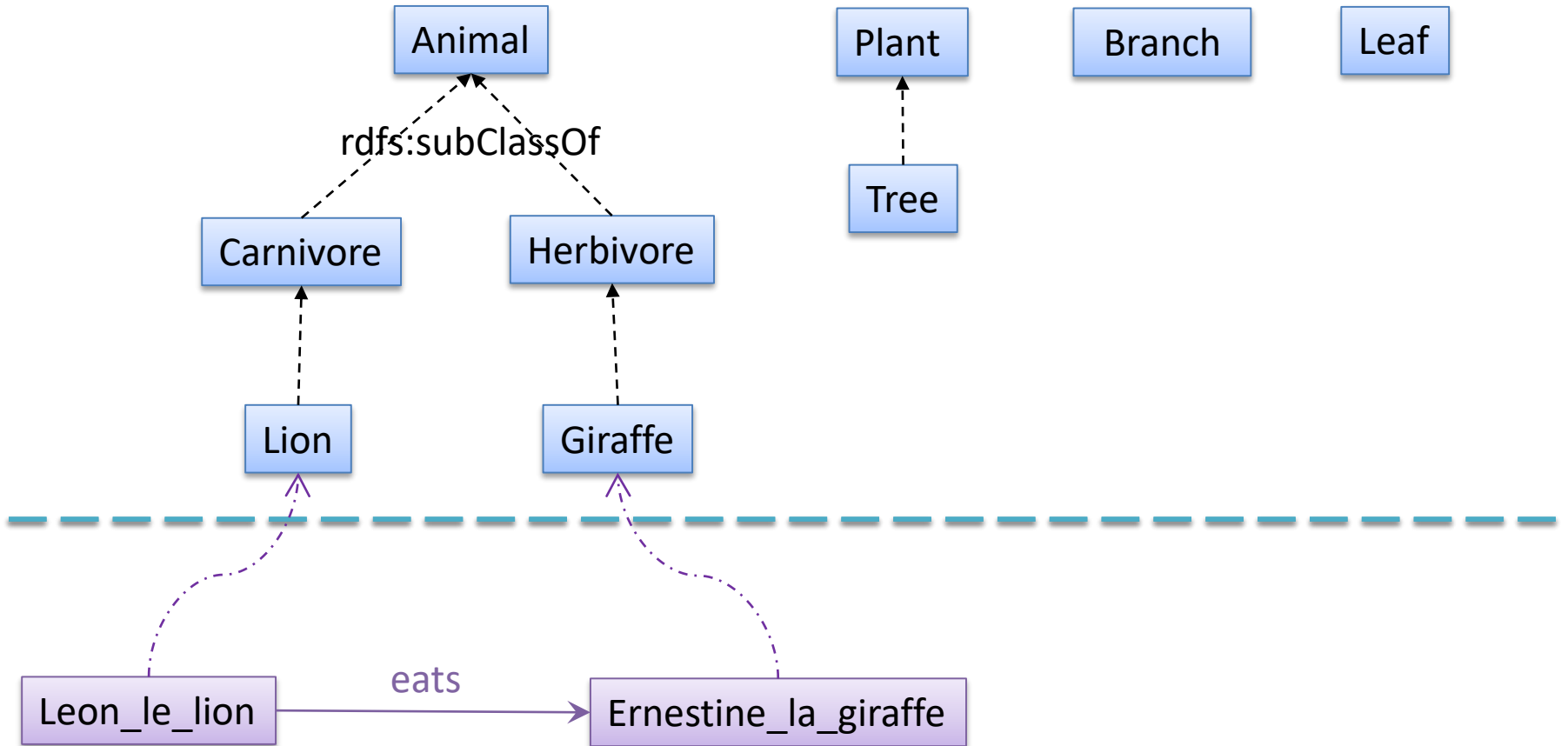


Disjonction entre plusieurs classes

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- **Les instances**
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Les instances



Les instances

RDF/XML Syntax

```
<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"/>
```

Les instances

Turtle Syntax

```
:Ernestine_la_girafe rdf:type owl:NamedIndividual ,  
                        :Giraffe .  
:Leon_le_lion rdf:type owl:NamedIndividual ,  
                       :Lion .
```

Plus simplement

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

Interprétation logique

Giraffe(Ernestine_la_girafe)

Lion(Leon_le_lion)

Identité des instances

- Pas d'égalité ni de distinction *a priori* entre les instances
- Il faut déclarer explicitement si les instances sont égales (constructeur [owl:sameAs](#)) ou distinctes (constructeurs [owl:differentFrom](#) et [owl:Alldifferent](#))

Identité des instances

RDF/XML Syntax

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

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

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

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

```
<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"/>
```

```
    <Lion rdf:about="#Leon_le_lion"/>
```

```
</owl:distinctMembers>
```

```
</owl:AllDifferent>
```

Identité des instances

Turtle Syntax

```
:Ernestine_la_girafe rdf:type :Giraffe .  
:Noemie rdf:type :Giraffe .  
:Gertrude rdf:type :Giraffe .  
:Leon_le_lion rdf:type :Lion .  
:Ernestine rdf:type :Giraffe .  
:Ernestine owl:sameAs :Ernestine_la_girafe .  
:Ernestine owl:differentFrom :Noemie .
```

Interprétation logique

$Giraffe(Ernestine) \wedge Giraffe(Ernestine_la_girafe) \wedge Giraffe(Noemie) \wedge$
 $sameAs(Ernestine, Ernestine_la_girafe) \wedge differentFrom(Ernestine, Noemie)$

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

Identité des instances

The screenshot displays a software interface for managing instances. On the left, a window titled "Individuals: Ernestine_la_girafe" contains a list of individuals: Ernestine, Ernestine_la_girafe (highlighted), Gertrude, Leon_le_lion, and Noemie. On the right, a window titled "Description: Ernestine_la_girafe" shows the following details:

- Annotations:** A section with a plus sign icon, currently empty.
- Types:** A section with a plus sign icon, showing the type "Giraffe" with a yellow circle icon and control buttons (question mark, at-sign, X, circle).
- Same Individual As:** A section with a plus sign icon, showing "Ernestine" with a purple diamond icon and control buttons.
- Different Individuals:** A section with a plus sign icon, showing "Gertrude, Leon_le_lion, Noemie" with a purple diamond icon and control buttons.

Identité des instances

Turtle Syntax

```
:Ernestine_la_girafe rdf:type :Giraffe .  
:Noemie rdf:type :Giraffe .  
:Gertrude rdf:type :Giraffe .  
:Leon_le_lion rdf:type :Lion .  
  
:Ernestine rdf:type :Giraffe .  
:Ernestine owl:sameAs :Ernestine_la_giraffe .  
:Ernestine owl:sameAs :Gertrude .  
:Ernestine owl:differentFrom :Noemie .
```



Qu'en pensez-vous ?

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



Identité des instances – Vérification de la cohérence

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- **Les descriptions de classes**
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Les descriptions de classes

- OWL distingue six types de descriptions de classes :
 - Identifiant de classe (URI/IRI)
 - intersection entre une ou plusieurs descriptions de classes
 - union entre une ou plusieurs descriptions de classes
 - complément d'une description de classe
 - énumération exhaustive de ses individus qui forment ses instances possibles
 - restriction de propriétés

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - **Par intersection**
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Description d'une classe par intersection *intersectionOf*

RDF/XML Syntax

```
<owl:Class rdf:ID="Omnivore">
```

```
<owl:equivalentClass>
```

Axiome de classe

```
<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>
```

Description d'une classe par intersection *intersectionOf*

Turtle Syntax

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

Interprétation logique

$\forall x \text{ Omnivore}(x) \leftrightarrow \text{Herbivore}(x) \wedge \text{Carnivore}(x)$ équivalence

Description d'une classe par intersection *intersectionOf*

Turtle Syntax

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

```
[] rdf:type owl:AllDisjointClasses ;  
  owl:members ( :Herbivore :Carnivore ) .
```

Qu'en pensez-vous ?





Description d'une classe par
intersection *intersectionOf* –
Vérification de la cohérence

Description d'une classe par intersection

Turtle Syntax

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

Axiome de classe

Turtle Syntax

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

*Quelle est d'après vous
la différence ?*

Interprétation logique ?





Description d'une classe par intersection

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - **Par union**
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Description d'une classe par union

unionOf

- Les carnivores sont définis comme l'union des canidés et des félidés
- Les canidés et les félidés sont des carnivores
- Les lions sont des félidés

Description des carnivores par union

RDF/XML Syntax

```
<owl:Class rdf:ID="Canide">
  <rdfs:subClassOf rdf:resource="#Carnivore"/>
</owl:Class>
<owl:Class rdf:ID="Felide">
  <rdfs:subClassOf rdf:resource="#Carnivore"/>
</owl:Class>
<owl:Class rdf:ID="Lion">
  <rdfs:subClassOf rdf:resource="#Felide"/>
</owl:Class>
<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>
```

Description des carnivores par union

Turtle Syntax

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

Interprétation logique

$\forall x \text{ Canide}(x) \rightarrow \text{Carnivore}(x)$

$\forall x \text{ Felide}(x) \rightarrow \text{Carnivore}(x)$

$\forall x \text{ Carnivore}(x) \leftrightarrow \text{Canide}(x) \vee \text{Felide}(x)$

Description des carnivores par union

The screenshot displays a software interface with two main panels. The left panel, titled 'Class hierarchy (inferred)', shows a tree structure of classes. The root is 'owl:Thing', which branches into 'Animal' and 'Plant'. 'Animal' further branches into 'Carnivore', 'Herbivore', 'Branch', 'Leaf', 'Les_girafes_copines', and 'Plant'. 'Carnivore' is highlighted in blue and has a sub-hierarchy including 'Canide', 'Felide', and 'Lion'. 'Herbivore' has sub-classes 'Giraffe' and 'Omnivore'. The right panel is divided into two sections: 'Annotations: Carnivore' and 'Description: Carnivore'. The 'Annotations' section is currently empty. The 'Description' section shows 'Equivalent To' as 'Canide or Felide' and 'SubClass Of' as 'Animal'. Both sections include icons for help, search, and other actions.

Class hierarchy (inferred)

Class hierarchy

Class hierarchy: Carnivore

Annotations Usage

Annotations: Carnivore

Annotations +

Description: Carnivore

Equivalent To +

Canide or Felide

SubClass Of +

Animal

Description des bouts de plante par union *unionOf*

- Les "bouts de plantes" sont définis comme l'union des plantes, des branches et des feuilles

Description des bouts de plante par union

Turtle Syntax

```
:PartOfPlant owl:equivalentClass [  
  rdf:type      owl:Class ;  
  owl:unionOf ( :Plant :Branch :Leaf )  
] .
```

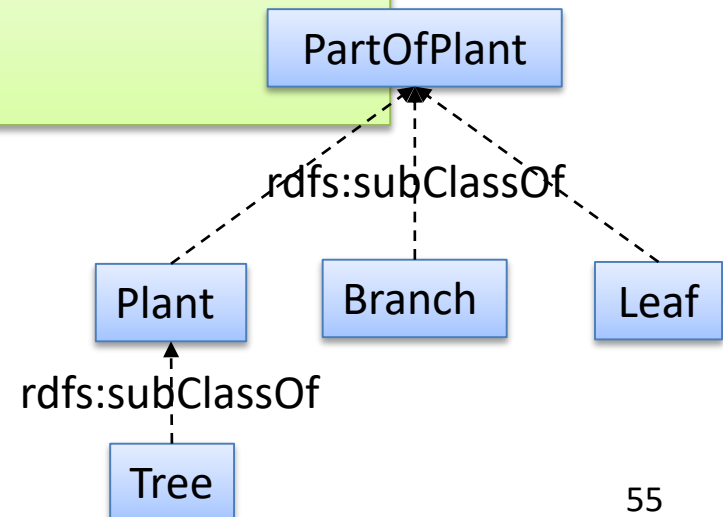
Interprétation logique

$\forall x \text{ PartOfPlant}(x) \rightarrow \text{Plant}(x) \vee \text{Branch}(x) \vee \text{Leaf}(x)$

$\forall x \text{ Plant}(x) \rightarrow \text{PartOfPlant}(x)$

$\forall x \text{ Branch}(x) \rightarrow \text{PartOfPlant}(x)$

$\forall x \text{ Leaf}(x) \rightarrow \text{PartOfPlant}(x)$



Description des bouts de plante par union

Turtle Syntax

```
:PartOfPlant rdfs:subClassOf [  
  rdf:type      owl:Class ;  
  owl:unionOf ( :Plant :Branch :Leaf )  
] .
```

*Que peut-on en déduire ?
Quelle est la différence
avec ce qui précède ?*



Interprétation logique ?

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - **Par complément**
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Description d'une classe par complément *complementOf*

- Les canidés sont des carnivores qui ne sont pas des félidés
- Les canidés sont des carnivores et ils ne sont pas des félidés

Description des canidés par complément

RDF/XML Syntax

```
<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>
```

Description des canidés par complément

Turtle Syntax

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

Interprétation logique

$$\forall x \text{ Canide}(x) \leftrightarrow \text{Carnivore}(x) \wedge \text{non Felide}(x)$$

Description des canidés par complément

The image shows a screenshot of a software interface for class management. On the left, a class hierarchy is displayed under the 'Classes' tab. The hierarchy starts with 'owl:Thing' at the top, followed by 'Animal'. Under 'Animal', there are two main branches: 'Carnivore' and 'Herbivore'. 'Carnivore' includes 'Omnivore', 'Felide', and 'Lion'. 'Herbivore' includes 'Omnivore' and 'Giraffe'. The 'Canide' class is highlighted in blue and is positioned between 'Lion' and 'Herbivore'. Other classes shown include 'Branch', 'Leaf', 'Plant', and 'Tree'. On the right, the 'Object properties' tab is active, showing the 'Description: Canide' section. This section contains three entries: 'Annotations' (with a plus sign), 'Equivalent To' (with a plus sign and a red circle around the text 'Carnivore and not Felide'), and 'SubClass Of' (with a plus sign and 'Carnivore' listed below). A 'General class axioms' section with a plus sign is also visible at the bottom.

Classes | Object properties

Class hierarchy: Cani ? [] [] [] []

owl:Thing

- Animal
 - Carnivore
 - Omnivore
 - Felide
 - Lion
 - Canide**
 - Herbivore
 - Omnivore
 - Giraffe
 - Branch
 - Leaf
 - Plant
 - Tree

Annotations: Canide

Annotations +

Description: Canide

Equivalent To +
Carnivore and not Felide

SubClass Of +
Carnivore

General class axioms +

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - **Par énumération**
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Description d'une classe par énumération *oneOf*

- Les girafes Ernestine_la_girafe, Noemie et Gertrude sont des girafes copines
- La classe des girafes copines est définie à partir de ses instances : Ernestine_la_girafe, Noemie et Gertrude

Description d'une classe par énumération *oneOf*

RDF/XML Syntax

```
<owl:Class rdf:ID="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>
```

Description d'une classe par énumération *oneOf*

Turtle Syntax

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

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

Interprétation logique

$\forall x \text{ Les_girafes_copines}(x) \leftrightarrow (x=\text{Ernestine_la_girafe}) \vee (x=\text{Noemie}) \vee (x=\text{Gertrude})$

Description d'une classe par énumération oneOf

Turtle Syntax

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

:Ernestine owl:sameAs :Ernestine_la_giraffe .

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

Que peut-on en déduire sur le type des trois instances Ernestine_la_girafe, Noemie et Gertrude ?

Et sur le type de l'instance Ernestine ?





Description d'une classe par
énumération *oneOf*

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
 - Disjonction entre plusieurs classes
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- **Les propriétés**
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

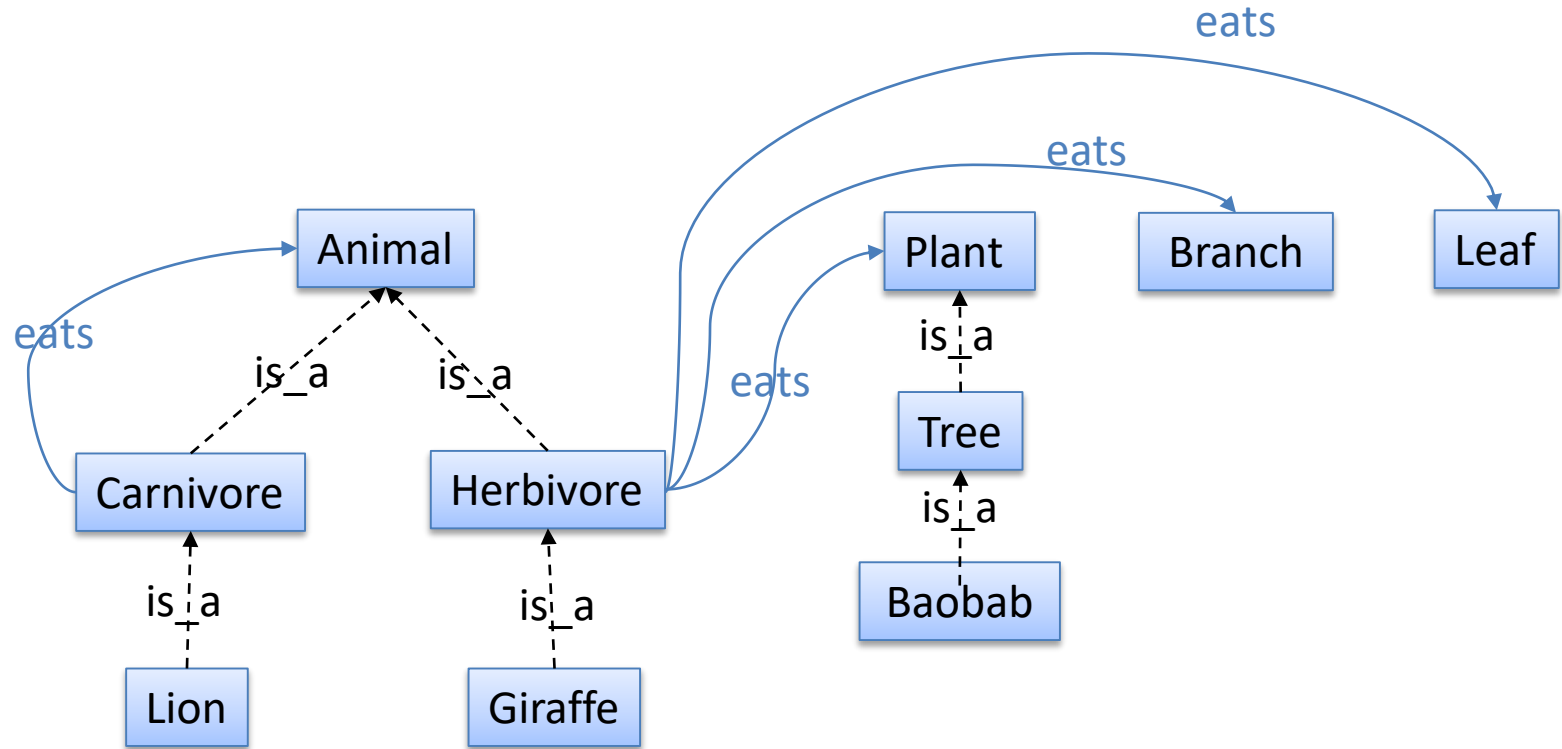
Les propriétés OWL

- *Properties can be used to state relationships between individuals or from individuals to data values.*

The African Wildlife Ontology

- les animaux carnivores **mangent** des animaux
- les animaux herbivores **mangent** des plantes ou des “bouts” de plantes

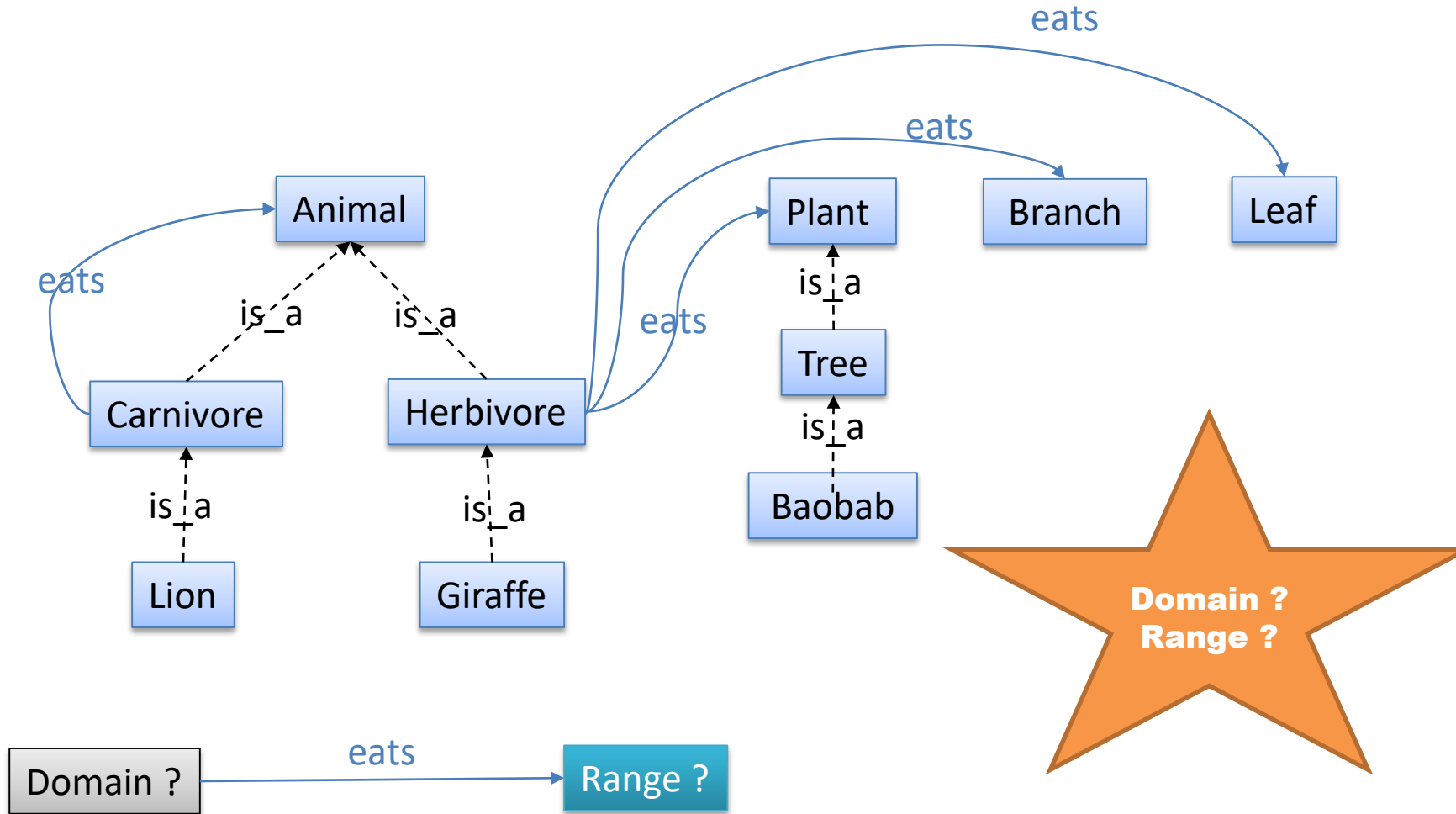
Les propriétés OWL



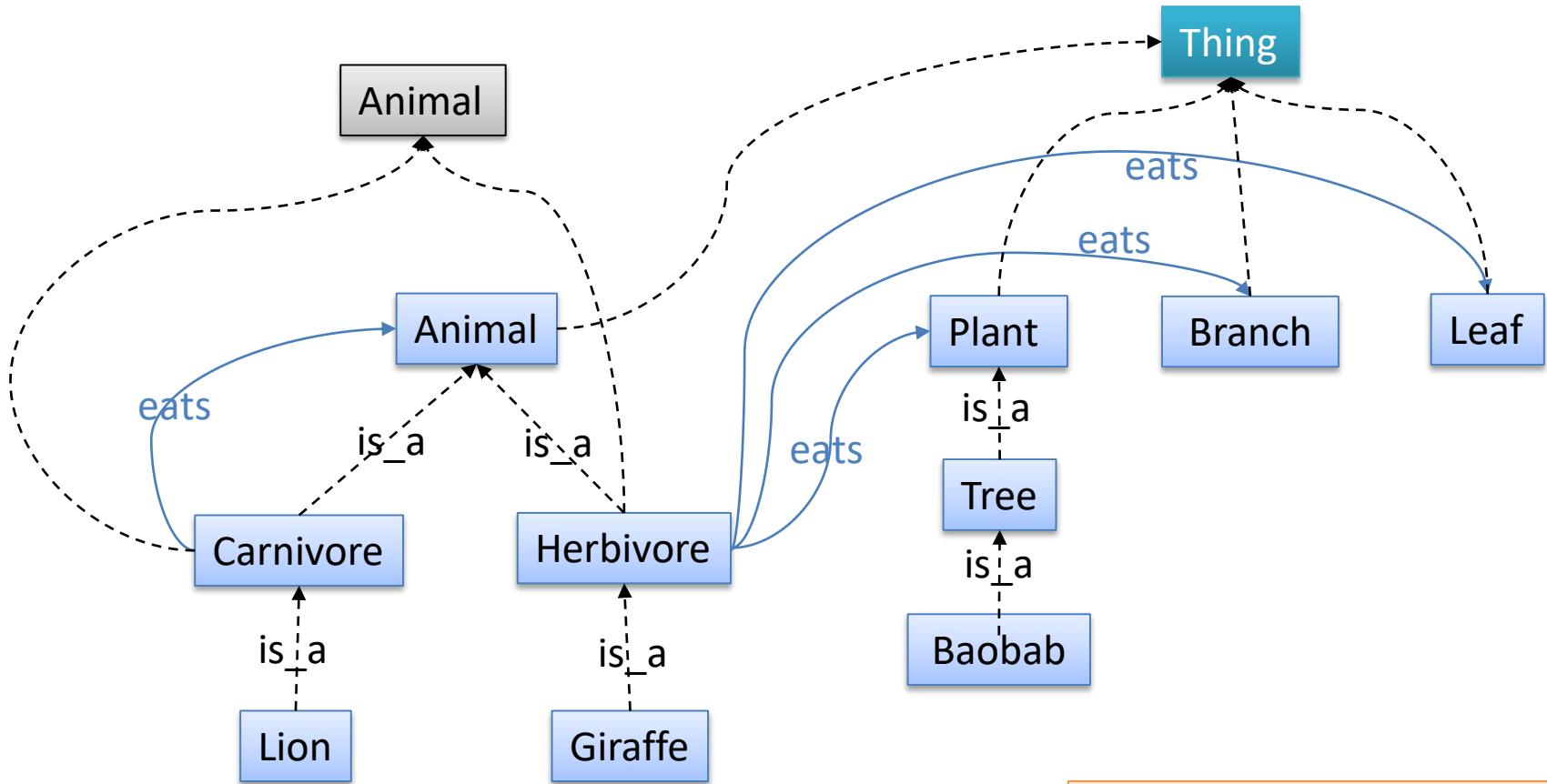
Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - **Les propriétés objets**
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Les propriétés objets



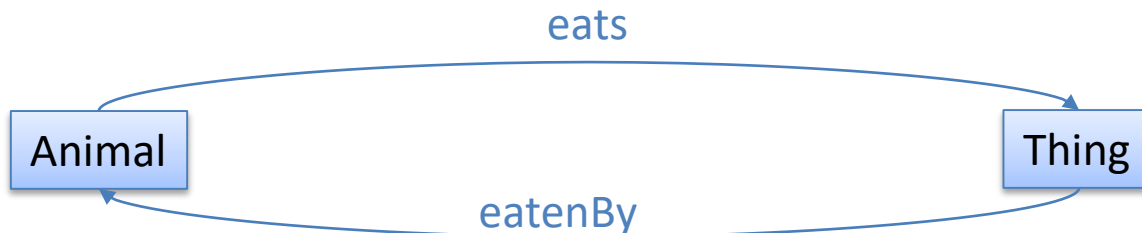
Les propriétés objets



Plus petit sur-type commun !

La propriété objet `eats`

- *ObjectProperty* (entre deux classes)
 - Object property **`eats`**
 - Domain: **`Animal`**
 - Range: **`Thing`**
- Propriété inverse



La propriété objet eats

RDF/XML Syntax

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

La propriété objet eats

Turtle Syntax

```
:eats      rdf:type owl:ObjectProperty ;  
           rdfs:domain  :Animal ;  
           rdfs:range  :Thing .  
:eatenBy   rdf:type owl:ObjectProperty ;  
           rdfs:domain  :Thing ;  
           rdfs:range   :Animal ;  
           owl:inverseOf :eats .
```

Inutile de préciser domain et range si Thing

Turtle Syntax simplifié

```
:eats      rdf:type owl:ObjectProperty ;  
           rdfs:domain  :Animal .  
:eatenBy   rdf:type owl:ObjectProperty ;  
           rdfs:range   :Animal ;  
           owl:inverseOf :eats .
```

La propriété objet eats

The screenshot displays a software interface for configuring an OWL object property. The main window is titled "Object property hierarchy: eatenBy" and shows a tree view with the following structure:

- owl:topObjectProperty
 - eatenBy
 - eats

The "eatenBy" property is selected, and its configuration is shown in the right-hand pane. The pane is titled "Annotations: eatenBy" and "Description: eatenBy".

The "Annotations" section is currently empty, with a "+" button to add annotations.

The "Description" section contains several options, each with a "+" button to add a description:

- Equivalent To
- SubProperty Of
- Inverse Of
 - eats**
- Domains (intersection)
- Ranges (intersection)
 - Animal**
- Disjoint With
- SuperProperty Of (Chain)

Below the "Description" section, there is a list of checkboxes for property characteristics:

- Functional
- Inverse functional
- Transitive
- Symmetric
- Asymmetric
- Reflexive
- Irreflexive

The "Asserted" dropdown menu is currently set to "Asserted".

Les sous-propriétés

La propriété objet eatsMeat

RDF/XML Syntax

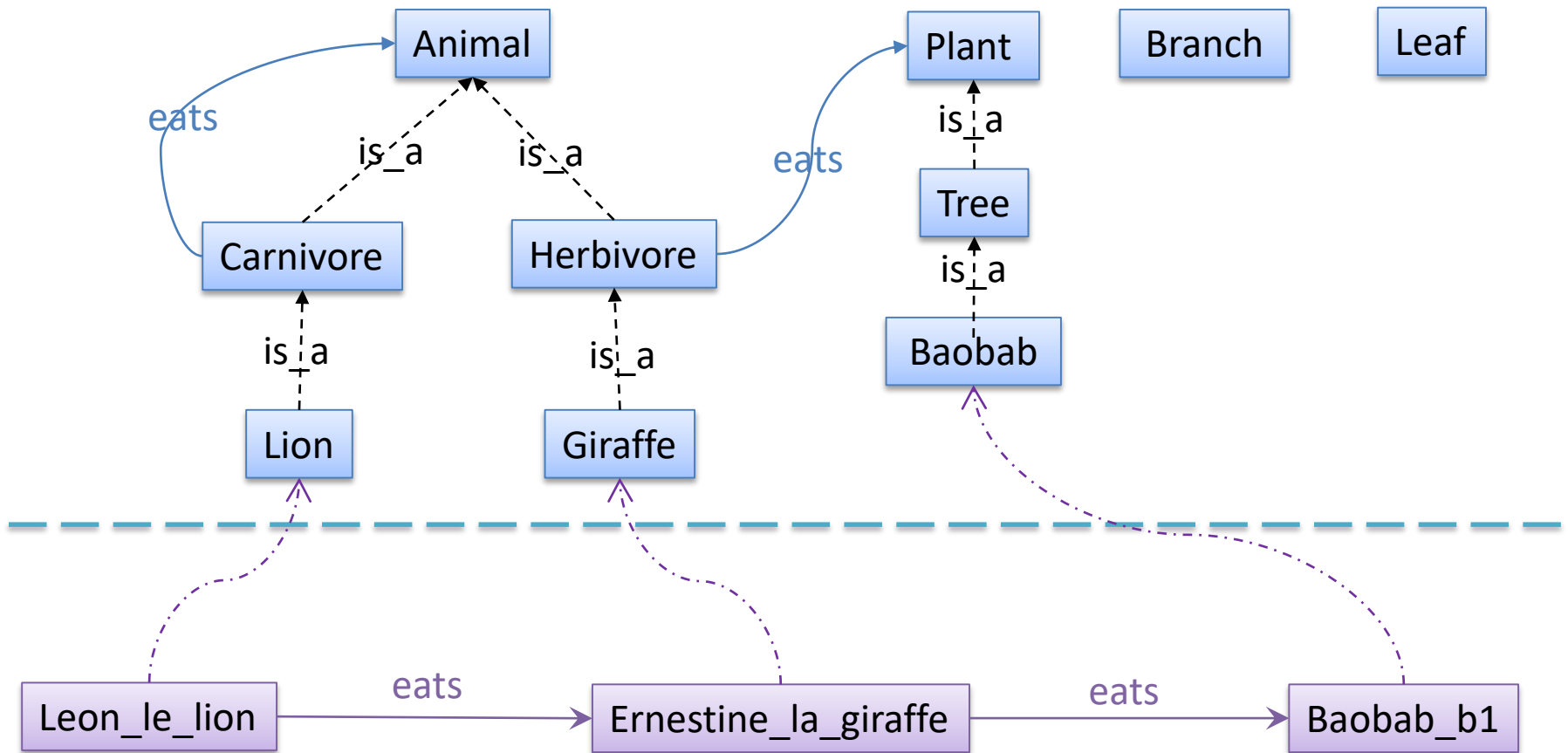
```
<owl:ObjectProperty rdf:about="eatsMeat">  
  <rdfs:subPropertyOf rdf:resource="#eats"/>  
  <rdfs:domain rdf:resource="#Carnivore "/>  
  <rdfs:range rdf:resource="#Animal"/>  
</owl:ObjectProperty>
```

Axiome de classe

Turtle Syntax

```
:eatsMeat    rdf:type owl:ObjectProperty ;  
             rdfs:domain    :Carnivore ;  
             rdfs:range     :Animal ;  
             rdfs:subPropertyOf :eats.
```

Les propriétés objets et les instances



Les propriétés objets et les instances

RDF/XML Syntax

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

Turtle Syntax

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

Les propriétés objets et les instances

The screenshot displays a Semantic Web browser interface with the following components:

- Individuals: Leon_le_lion** (top left): A list of individuals including Ernestine, Ernestine_la_girafe, Gertrude, **Leon_le_lion** (selected), and Noemie.
- Annotations** (top middle): A section for annotations, currently empty.
- Description: Leon_le_lion** (middle left):
 - Types**: Shows the type **Lion**.
 - Same Individual As**: A section for identifying other individuals.
 - Different Individuals**: Shows that Leon_le_lion is different from Ernestine_la_girafe, Gertrude, and Noemie.
- Property assertions: Leon_le_lion** (middle right):
 - Object property assertions**: Shows the assertion **eats Ernestine_la_girafe**.
 - Data property assertions**: A section for data property assertions.
 - Negative object property assertions**: A section for negative object property assertions.
 - Negative data property assertions**: A section for negative data property assertions.

Les propriétés objets et les instances

Turtle Syntax

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

Complétez le code pour:

Ernestine_la_giraffe → eats → Baobab_b1



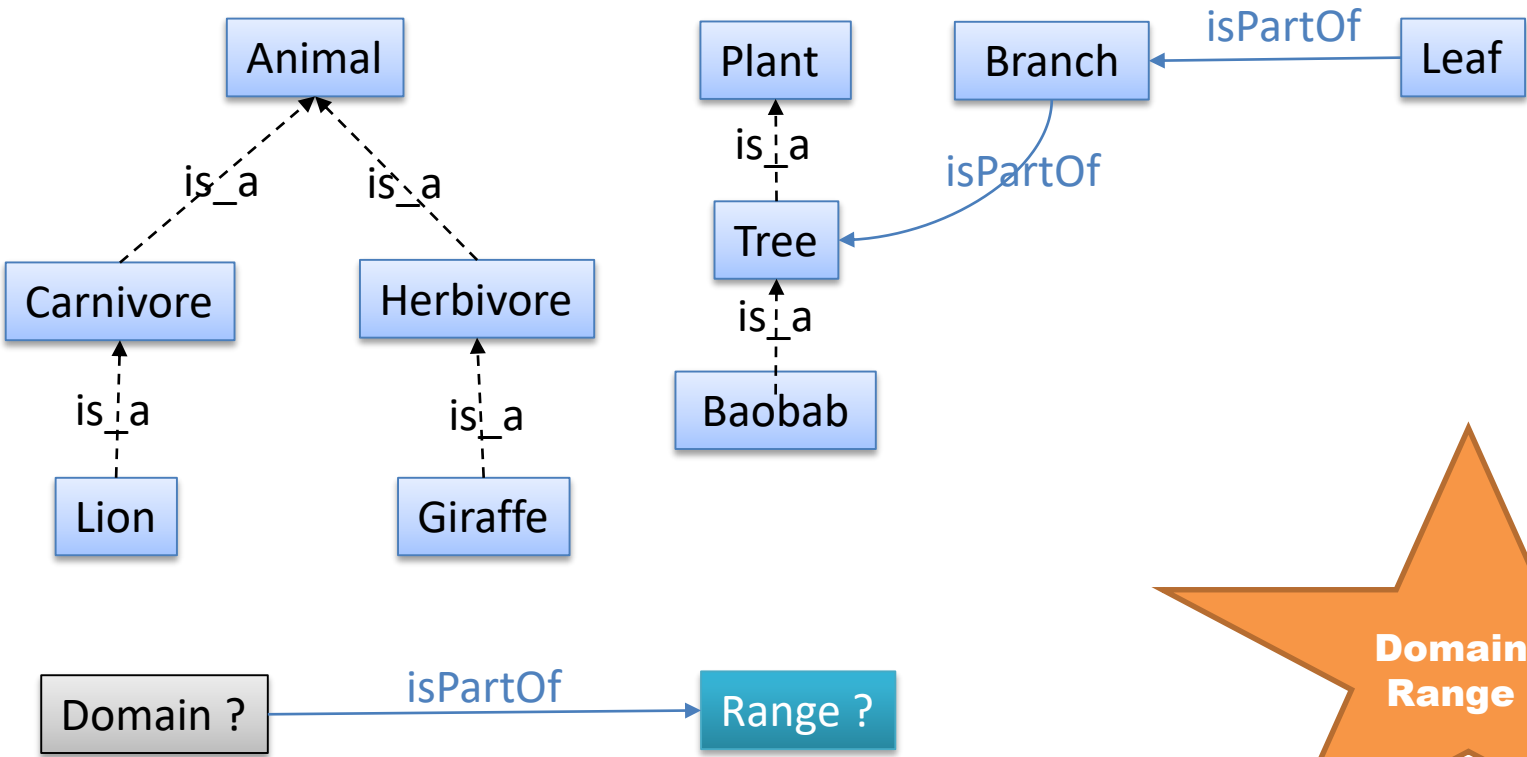


Les propriétés objets et les instances

Les propriétés objets

- The African Wildlife Ontology
 - les animaux carnivores mangent des animaux
 - les animaux herbivores mangent des plantes ou des “**bouts**” de plantes

La propriété objet ispartOf





La propriété objet `ispartOf`

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - **Les propriétés de données**
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Les propriétés de données

- Comment attribuer un âge à un animal?



La propriété de données `hasAge`

- *[DatatypeProperty](#)* (entre une classe et un « datatype »)
 - Datatype property **`hasAge`**
 - Domain: **`Thing`**
 - Range: **`nonNegativeInteger`** (**`Datatype`**)

La propriété de données hasAge

RDF/XML Syntax

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

Turtle Syntax

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

Les propriétés de données et les instances

- Supposons qu'Ernestine_la_girafe a 6 ans

RDF/XML Syntax

```
<Giraffe rdf:about="Ernestine_la_girafe">
  <hasAge
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">6<hasAge/>
</Giraffe>
```

Turtle Syntax

```
:Ernestine_la_girafe rdf:type :Giraffe ;
                      :hasAge "6"^^xsd:nonNegativeInteger .
```

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - **Les caractéristiques des propriétés**
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Les caractéristiques des propriétés

The screenshot displays a software interface for managing OWL object properties. On the left, a tree view shows the hierarchy: owl:topObjectProperty > eatenBy > eats. The main area is divided into three panels:

- Annotations: eatenBy**: Shows a list of annotations with a plus sign to add more.
- Description: eatenBy**: Shows a list of domain and range relationships, including:
 - Equivalent To (+)
 - SubProperty Of (+)
 - Inverse Of (+)
 - eats** (indicated by a blue bar)
 - Domains (intersection) (+)
 - Ranges (intersection) (+)
 - Animal** (indicated by a yellow dot)
 - Disjoint With (+)
 - SuperProperty Of (Chain) (+)
- Characteristics: eatenBy**: A list of checkboxes for property characteristics, which is circled in red:
 - Functional
 - Inverse functional
 - Transitive
 - Symmetric
 - Asymmetric
 - Reflexive
 - Irreflexive

Les propriétés fonctionnelles – la propriété hasAge

Active Ontology × Entities × Classes × Object Properties × Data Properties × Annotation Properties × Individuals by class × OWLViz × DL Query × SPARQL Query ×

Data property hierarchy: hasAge

Annotations Usage

Annotations: hasAge

Annotations +

Tout animal a au plus un âge !

Characteristics: hasAge

Functional

Propriété **fonctionnelle** : propriété qui a au plus une valeur

Description: hasAge

Equivalent To +

SubProperty Of +

Domains (intersection) +

Ranges +

xsd:nonNegativeInteger

Disjoint With +

Les propriétés fonctionnelles – la propriété hasAge

RDF/XML Syntax

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

Turtle Syntax

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

Les propriétés fonctionnelles – la propriété hasAge

Turtle Syntax

```
:hasAge      rdf:type      owl:DatatypeProperty ;  
             rdf:type      owl:FunctionalProperty ;  
             rdfs:range     xsd:nonNegativeInteger .  
  
:Ernestine_la_girafe rdf:type :Giraffe ;  
                    :hasAge "5"^^xsd:nonNegativeInteger .  
  
:Ernestine  rdf:type :Giraffe ;  
            owl:sameAs :Ernestine_la_girafe ;  
            :hasAge "4"^^xsd:nonNegativeInteger .  
  
:Gertrude :hasAge "8"^^xsd:nonNegativeInteger .
```

Qu'en pensez-vous ?





Les propriétés fonctionnelles – la propriété `hasAge`

Les caractéristiques des propriétés

- 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*)

$$P(y, x) \wedge P(z, x) \rightarrow y = z$$

Les propriétés transitives – la propriété `ispartOf`

Turtle Syntax

```
:isPartOf      rdf:type owl:ObjectProperty ;  
               rdf:type owl:TransitiveProperty .  
  
:Baobab        rdfs:subClassOf :Tree .  
  
:Baobab_b1     rdf:type :Baobab .  
  
:Branch_baobab_b1  rdf:type :Branch ;  
                   :ispartOf :Baobab_b1 .  
  
:Leaf_baobab_l1  rdf:type :Leaf ;  
                   :ispartOf :Branch_baobab_b1 .
```

Que peut-on en déduire ?





Les propriétés transitives – la propriété `ispartOf`

Les caractéristiques des propriétés

- 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. *isPartOf*)

$$P(x, y) \wedge P(y, z) \rightarrow P(x, z)$$

Les propriétés symétriques— la propriété hasSameAge

Turtle Syntax

```
:hasSameAge  rdf:type      owl:ObjectProperty ;  
              rdf:type      owl:SymmetricProperty;  
              rdfs:domain   :Animal ;  
              rdfs:range    :Animal .  
  
:Gertrude   rdf:type :Giraffe .  
  
:Ernestine_la_giraffe  rdf:type :Giraffe ;  
                        :hasSameAge :Gertrude .
```

Que peut-on en déduire ?





Les propriétés symétriques—
la propriété `hasSameAge`

Les propriétés asymétriques – la propriété `isOlderThan`

Turtle Syntax

```
:isOlderThan rdf:type      owl:ObjectProperty ;  
             rdf:type      owl:AsymmetricProperty ;  
             rdfs:domain   :Animal ;  
             rdfs:range    :Animal .  
  
:Gertrude    rdf:type :Giraffe ;  
             :isOlderThan :Ernestine_la_giraffe .  
  
:Ernestine_la_giraffe rdf:type :Giraffe ;  
                      :isOlderThan :Gertrude .
```

Qu'en pensez-vous ?





Les propriétés asymétriques—
la propriété `isOlderThan`

Les caractéristiques des propriétés

- 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. *isPartOf*)

$$P(x, y) \wedge P(y, z) \rightarrow P(x, z)$$

- Symétrique (e.g. *hasSameAge*)

$$P(x, y) \Leftrightarrow P(y, x)$$

- Asymétrique (OWL2) (e.g. *isOlderThan*)

$$P(x, y) \wedge P(y, x) \rightarrow \perp$$

Les propriétés réflexives – la propriété hasSameAge

Turtle Syntax

```
:hasSameAge  rdf:type      owl:ObjectProperty ;  
              rdf:type      owl:ReflexiveProperty ;  
              rdfs:domain    :Animal ;  
              rdfs:range     :Animal .  
  
:Gertrude    rdf:type :Giraffe .  
:Ernestine_la_girafe rdf:type :Giraffe .
```

Que peut-on en déduire ?





Les propriétés réflexives – la propriété `hasSameAge`

Les propriétés irreflexives – la propriété `isOlderThan`

Turtle Syntax

```
:isOlderThan      rdf:type      owl:ObjectProperty ;  
                  rdf:type      owl:IrreflexiveProperty ;  
                  rdfs:domain    :Animal ;  
                  rdfs:range     :Animal .  
  
:Ernestine_la_girafe  rdf:type :Giraffe ;  
                      :isOlderThan :Ernestine .  
  
:Ernestine  rdf:type :Giraffe ;  
            owl:sameAs :Ernestine_la_girafe .
```

Que peut-on en déduire ?





Les propriétés irreflexives –
la propriété `isOlderThan`

Les caractéristiques des propriétés

- 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. *isPartOf*)
 $P(x, y) \wedge P(y, z) \rightarrow P(x, z)$
- Symétrique (e.g. *hasSameAge*)
 $P(x, y) \Leftrightarrow P(y, x)$
- Asymétrique (OWL2) (e.g. *isOlderThan*)
 $P(x, y) \wedge P(y, x) \rightarrow \perp$
- Réflexive (OWL2) (e.g. *hasSameAge*)
 $P(x, x)$
- Irréflexive (OWL2) (e.g. *isOlderThan*)
non $P(x, x)$

Résumé sur les propriétés et leurs constructeurs

- Les constructeurs de RDF Schema :
`rdfs:subPropertyOf`, `rdfs:domain` **et** `rdfs:range`
- Les relations aux autres propriétés:
`owl:equivalentProperty` **et** `owl:inverseOf`
- Leurs caractéristiques
 - Les contraintes de cardinalités: `owl:FunctionalProperty` **et** `owl:InverseFunctionalProperty`
 - Les caractéristiques logiques :
`owl:SymmetricProperty` **et** `owl:TransitiveProperty`
 - OWL2 : `owl:AsymmetricProperty`, `owl:ReflexiveProperty`,
`owl:IrreflexiveProperty`, `owl:propertyDisjointWith`

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- **Les restrictions de propriétés**
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Rappel: les descriptions de classes

OWL (cf. Diapositive 42)

- OWL distingue six types de descriptions de classes :
 - Identifiant de classe (URI)
 - intersection entre une ou plusieurs descriptions de classes
 - union entre une ou plusieurs descriptions de classes
 - complément d'une description de classe
 - énumération exhaustive de ses individus qui forment ses instances possibles
 - **restriction de propriétés**

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - **Les restrictions de valeurs**
 - Les restrictions de cardinalités
- Les spécificités OWL2
- Les règles

Les restrictions de valeurs

- Supposons qu'`Ernestine_la_girafe` mange `Leon_le_lion`
- Si on lance le raisonneur de Protégé:
 - Aucune incohérence n'est détectée !
 - Pourtant on voudrait pouvoir dire que les herbivores ne mangent que des plantes...
- Utilisation des restrictions de valeurs
 - **`allvaluesFrom`**
 - `someValuesFrom`
 - `hasValue`

La restriction de valeurs `allvaluesFrom`

- The African Wildlife Ontology
 - les animaux herbivores sont des animaux qui mangent des plantes ou des “bouts” de plantes
 - “Herbivores are exactly those animals that eat **only** plants or parts of plants.”*
- Comment définir les animaux herbivores?

La restriction de valeurs `allvaluesFrom`

- Les herbivores sont des animaux: `subclassOf`
Les herbivores sont des animaux qui ne mangent que des plantes

« *Les herbivores mangent **uniquement** des plantes* »

- La classe `Herbivore` est redéfinie avec une restriction sur le `range` de sa propriété `eats`
- Utilisation de la restriction `allvaluesFrom`

La restriction de valeurs `allvaluesFrom`

RDF/XML Syntax

```
<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>
```

La restriction de valeurs `allvaluesFrom`

Turtle syntax

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

```
  rdfs:comment "Herbivores are exactly those animals  
that eat only plants or parts of plants."^^xsd:string .
```



Les restrictions de valeurs sont toujours définies à partir d'une classe sur une propriété

La restriction de valeurs `allvaluesFrom`

Turtle syntax

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

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

```
:Leon_le_lion rdf:type :Lion .
```

```
:Ernestine_la_girafe rdf:type :Giraffe ;  
  :eats :Leon_le_lion .
```

Qu'en pensez-vous?





La restriction de valeurs `allvaluesFrom`

La restriction de valeurs `allvaluesFrom`

- Supposons maintenant
qu'Ernestine_la_girafe mange
Leon_le_lion sachant qu'aucune classe
n'est disjointe

La restriction de valeurs `allvaluesFrom`

- Vérification de la cohérence

- Nouvelles connaissances inférées

```
Ernestine_la_giraffe is-a Giraffe
```

```
Ernestine_la_giraffe is-a Herbivore
```

```
Leon_le_lion is-a Lion
```

```
Ernestine_la_giraffe eats Leon_le_lion
```

```
Ernestine_la_giraffe eats only Plant
```



```
Leon_le_lion is-a Plant
```

```
Leon_le_lion eatenBy Ernestine_la_giraffe
```

- Possible car les classes `Lion` et `Plant` ne sont pas disjointes !

La restriction de valeurs `allvaluesFrom`

- Vérification de la cohérence

The screenshot displays a Semantic Web editor interface with several panels:

- Class hierarchy (inferred):** Shows a tree structure starting with `owl:Thing`, containing `Animal`, `Branch`, `Leaf`, `Les_girafes_copines`, and `Plant`. The `Animal` class is highlighted.
- Individuals: Leon_le_lion:** Lists individuals: `Ernestine`, `Ernestine_la_girafe`, `Gertrude`, `Leon_le_lion` (selected), and `Noemie`.
- Annotations: Leon_le_lion:** Shows a list of annotations for the selected individual.
- Description: Leon_le_lion:** Shows the class hierarchy for the selected individual. The `Lion` class is highlighted and circled in red, indicating a restriction. The `Plant` class is also listed below it.
- Property assertions: Leon_le_lion:** Shows object property assertions for the selected individual, including `eaten-by Ernestine_la_girafe` and `eaten-by Ernestine`.

La restriction de valeurs `allValuesFrom`

- Les herbivores sont des animaux qui mangent uniquement des plantes ou des “bouts” de plantes
- Supposons que les “bouts” de plantes sont :
 - des branches
 - **ou** des feuilles
 - **ou** des bouts de plantes (propriété `isPartOf`)

Définition de la classe bouts de plantes par restriction de valeurs

Turtle syntax

```
:PartOfPlant owl:equivalentClass [
  rdf:type      owl:Class ;
  owl:unionOf (:Branch :Leaf
                [ rdf:type owl:Restriction ;
                  owl:onProperty :isPartOf ;
                  owl:allValuesFrom :Plant ]
                )
] .
```

Définition de la classe bouts de plantes par restriction de valeurs

The image shows a software interface for defining classes. On the left, a class hierarchy is displayed under the 'Classes' tab. The hierarchy starts with 'owl:Thing' and includes 'Animal', 'Branch', 'Leaf', 'Les girafes copines', 'PartOfPlant', and 'Plant'. 'PartOfPlant' is highlighted with a blue background. Above the hierarchy are icons for adding and removing classes, and a dropdown menu set to 'Asserted'. On the right, the 'Object properties' tab is active, showing the 'Description: PartOfPlant'. The description is defined as 'Equivalent To' a disjunction: 'Branch or Leaf or (isPartOf only Plant)'. Other expandable sections for 'Annotations', 'SubClass Of', and 'General class axioms' are visible but empty.

Classes | Object properties

Class hierarchy: PartOfPlant

owl:Thing

- Animal
- Branch
- Leaf
- Les girafes copines
- PartOfPlant**
- Plant

Annotations +

Description: PartOfPlant

Equivalent To +

● Branch or Leaf or (isPartOf only Plant)

SubClass Of +

General class axioms +

Définition des herbivores par restriction de valeurs

- Les herbivores sont des animaux qui mangent uniquement des plantes ou des “bouts” de plantes
- Définissons maintenant la classe Herbivore à partir de la classe `PartOfPlant`

Définition des herbivores par restriction de valeurs

Turtle syntax

```
:Herbivore rdfs:subClassOf [  
  owl:unionOf ([ rdf:type owl:Restriction ;  
    owl:onProperty :eats ;  
    owl:allValuesFrom :Plant  
  ]  
  [ rdf:type owl:Restriction ;  
    owl:onProperty :eats ;  
    owl:allValuesFrom :PartOfPlant  
  ]  
)  
] .
```

Définition des herbivores par restriction de valeurs

RDF/XML syntax

```
<owl:Class rdf:about="#Herbivore">
  <rdfs: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 rdf:resource="#PartOfPlant"/>
        </owl:Restriction>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```


Les restrictions de valeurs

- Les restrictions de valeurs sont définies à partir d'une classe sur une propriété
 - `allvaluesFrom` (uniquement)
 - **`someValuesFrom`**
 - `hasValue`

La restriction de valeurs `someValuesFrom`

- The African Wildlife Ontology
 - les animaux carnivores sont des animaux qui mangent des animaux

“Carnivores are exactly those animals that eat also animals.”
- Comment définir les animaux carnivores?

La restriction de valeurs `someValuesFrom`

- Les carnivores sont des animaux: `subClassOf`
Les carnivores sont des animaux qui mangent des animaux, mais pas seulement

« *Les carnivores mangent **notamment** des animaux* »

- La classe `Carnivore` est redéfinie avec une restriction sur le `range` de sa propriété `eats`
- Utilisation de la restriction `someValuesFrom`



≠ *AllValuesFrom* : **uniquement**

Définition des carnivores par restriction de valeurs

RDF/XMLsyntax

```
<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>
```

Définition des carnivores par restrictions de valeurs

Turtle syntax

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

```
    rdfs:comment "Carnivores are exactly those  
animals that eat also animals."^^xsd:string .
```



Les restrictions de valeurs sont toujours définies à partir d'une classe sur une propriété

Définition des carnivores par restrictions de valeurs

The screenshot displays a Semantic Web editor interface with two main panels. The left panel, titled 'Class hierarchy: Carnivore', shows a tree structure of classes. The right panel, titled 'Annotations: Carnivore' and 'Description: Carnivore', shows the class's properties and restrictions.

Class hierarchy:

- owl:Thing
 - Animal
 - Carnivore**
 - Canide
 - Felide
 - Lion
 - Omnivore
 - Herbivore
 - Branch
 - Leaf
 - Les_girafes_copines
 - Plant

Annotations: Carnivore

Annotations +

Description: Carnivore

Equivalent To +
● Canide or Felide

SubClass Of +
● Animal
● eats some Animal

General class axioms +

SubClass Of (Anonymous Ancestor)

La restriction de valeurs `someValuesFrom`

Turtle syntax

```
:eats rdf:type owl:ObjectProperty ;
      rdfs:domain :Animal .

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

:Baobab rdfs:subClassOf :Tree .

:Baobab_b1 rdf:type :Baobab .
:Leon_le_lion rdf:type :Lion ;
               :eats :Baobab_b1 .
```

Qu'en pensez-vous?



Les restrictions de valeurs

- Les restrictions de valeurs sont définies à partir d'une classe sur une propriété
 - `allvaluesFrom` (uniquement)
 - `someValuesFrom` (notamment)
 - **`hasValue`**

La restriction de valeurs `hasValue`

- Les herbivores « basilicéens » sont des herbivores qui ne mangent que du basilic, le basilic étant une **instance** de la classe `Plant`
- Les basilicéens sont des herbivores : `subclassOf`
- La classe `Basilicéen` est définie comme une sous-classe de la classe `Herbivore` avec une restriction sur le `range` de sa propriété `eats`
- Utilisation de la restriction `hasValue`



*≠ `AllValuesFrom` : car le range est restreint à une **instance** et non une sous-classe*

Définition des basilicéen par restriction de valeurs

RDF/XML Syntax

```
<Plant rdf:about="Basilic"/>
```

```
<owl:Class rdf:ID="Herbivore_Basiliceen">  
  <rdfs:subClassOf rdf:resource=#Herbivore/>  
  <rdfs:subClassOf>  
    </owl:Class>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="eats"/>  
      <owl:hasValue rdf:resource="Basilic"/>  
    </owl:Restriction>  
  </owl:Class>  
</rdfs:subClassOf>  
</owl:Class>
```

Définition des basilicéens par restriction de valeurs

Turtle Syntax

```
:Basilic      rdf:type      :Plant .  
:Herbivore_Basiliceen  
      rdfs:subClassOf  :Herbivore ;  
      rdfs:subClassOf  [  
          rdf:type owl:Restriction ;  
          owl:onProperty :eats ;  
          owl:hasValue :Basilic  
      ] .
```

Définition des basilicéens par restriction de valeurs

The screenshot shows a class hierarchy editor with the following structure:

- owl:Thing
 - PartOfPlant
 - Animal
 - Carnivore
 - Lion
 - Herbivore
 - Herbivore_Basilicéen (highlighted)
 - Giraffe
 - Branch
 - Leaf
 - Plant
 - Tree

The right-hand pane displays the details for the selected class, **Herbivore_Basilicéen**:

- Annotations: +
- Description: Herbivore_Basilicéen
- Equivalent To: +
- SubClass Of: +
 - eats value Basilic
 - Herbivore
- General class axioms: +
- SubClass Of (Anonymous Ancestor):
 - (eats only Plant) or (eats only PartOfPlant)

La restriction de valeurs `hasValue`

- Les « copines d'Ernestine » sont des girafes qui sont des copines d'`Ernestine_la_girafe`.
- La propriété `isFriendOf` permet de déclarer que deux individus (au sens large) sont amis. Quel est son domaine, son range ? Précisez ses caractéristiques : fonctionnelle, transitive, symétrique, réflexive?





Définition de la propriété `isFriendOf`

La restriction de valeurs `hasValue`

- Les « copines d'Ernestine » sont des girafes qui sont des copines d'`Ernestine_la_girafe`.
- Définir maintenant la classe `copines_d_Ernestine`





Définition de la classe

`Copines_d_Ernestine` **par**
restriction de valeurs

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - **Les restrictions de cardinalités**
- Les spécificités OWL2
- Les règles

Les restrictions de cardinalités

- The African Wildlife Ontology
 - Tout animal a un âge
- Comment définir cette contrainte ?

Les restrictions de cardinalités

- The African Wildlife Ontology
 - Tout animal a un âge
- Utilisation des contraintes de cardinalité
 - `minCardinality` (au moins)
 - `maxCardinality` (au plus)
 - `cardinality(exactement)`

Les restrictions de cardinalités

RDF/XML Syntax

```
<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>
```

Les restrictions de cardinalités

Turtle syntax

```
:Animal rdfs:subClassOf [  
  rdf:type owl:Restriction ;  
  owl:onProperty :hasAge ;  
owl:cardinality "1"^^xsd:nonNegativeInteger ;  
  owl:onDataRange xsd:nonNegativeInteger  
] .
```

```
:Animal rdfs:subClassOf [  
  rdf:type owl:Restriction ;  
  owl:onProperty :hasAge ;  
  owl:minCardinality "1"^^xsd:nonNegativeInteger ;  
  owl:onDataRange xsd:nonNegativeInteger  
] .
```



La propriété `hasAge` est une propriété fonctionnelle

Les restrictions de cardinalités

Turtle syntax

```
:Animal rdfs:subClassOf [  
  rdf:type owl:Restriction ;  
  owl:onProperty :hasAge ;  
  owl:minCardinality "1"^^xsd:nonNegativeInteger ;  
  owl:onDataRange xsd:nonNegativeInteger  
] .
```

Pensez-vous que cette nouvelle restriction aura une influence sur les instances?





Les restrictions de cardinalités – vérification de la cohérence

Les restrictions de propriétés - Exercices

- Définissez :
 - Les girafes sont des herbivores qui ne mangent que des feuilles.
 - Les lions sont des carnivores qui mangent notamment des herbivores.
 - Les lions ont au plus un copain qui ne peut être qu'un lion.
 - Les branches sont nécessairement des bouts d'arbre.
 - Les feuilles sont nécessairement des bouts de branches.
 - Un arbre a au moins une branche.





Les restrictions de propriétés - Exercices

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- **Les spécificités OWL2**
- Les règles

Les spécificités avancées de OWL2

- Les chaînes de propriétés
- Les clés
- Définitions d'annotations plus riches
- Définitions de types de données plus riches
- Définitions de restrictions sur les cardinalités plus avancées

Les spécificités avancées de OWL2

- Les 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>
```

Turtle Syntax

```
:hasGrandparent owl:propertyChainAxiom ( :hasParent :hasParent ) .
```

Les spécificités avancées de OWL2

- Les clés

```
<owl:Class rdf:about="#Animal">  
  <owl:hasKey rdf:parseType="Collection">  
    <owl:ObjectProperty rdf:about="hasTatouage"/>  
  </owl:hasKey>  
</owl:Class>
```

Turtle Syntax

```
:Animal owl:hasKey ( :hasTatouage ) .
```


Les spécificités avancées de OWL2

- Les annotations

```
<owl:Class rdf:ID="Carnivore">
  <rdfs:subClassOf rdf:resource="Animal"/>
</owl:Class>
<owl:Axiom>
  <owl:annotatedSource rdf:resource="Carnivore"/>
  <owl:annotatedProperty rdf:resource="&rdfs;subClassOf"/>
  <owl:annotatedTarget rdf:resource="Animal"/>
  <rdfs:comment>States that a carnivore is an animal.
  </rdfs:comment>
</owl:Axiom>
```

Les spécificités avancées de OWL2

- Les annotations

Turtle Syntax

```
:Carnivore rdfs:subClassOf :Animal .  
[] rdf:type owl:Axiom ;  
    owl:annotatedSource :Carnivore ;  
    owl:annotatedProperty rdfs:subClassOf ;  
    owl:annotatedTarget :Animal ;  
    rdfs:comment "States that a carnivore  
is an animal."^^xsd:string .
```

Les spécificités avancées de OWL2

- Définitions de types de données plus riches
(e.g. owl:NegativePropertyAssertion)
Exemple : Ernestine ne peut pas avoir 12 ans
- Définitions de restrictions sur les cardinalités plus avancées
(e.g. owl:maxQualifiedCardinality, owl:minQualifiedCardinality)
Exemple : Léon a au moins 2 copines qui sont des copines d'Ernestines

OWL Lite

2.1 OWL Lite Synopsis

The list of OWL Lite language constructs is given below.

RDF Schema Features:

- [*Class \(Thing, Nothing\)*](#)
- [*rdfs:subClassOf*](#)
- [*rdf:Property*](#)
- [*rdfs:subPropertyOf*](#)
- [*rdfs:domain*](#)
- [*rdfs:range*](#)
- [*Individual*](#)

Property Restrictions:

- [*Restriction*](#)
- [*onProperty*](#)
- [*allValuesFrom*](#)
- [*someValuesFrom*](#)

Class Intersection:

- [*intersectionOf*](#)

Datatypes

- [*xsd datatypes*](#)

(In)Equality:

- [*equivalentClass*](#)
- [*equivalentProperty*](#)
- [*sameAs*](#)
- [*differentFrom*](#)
- [*AllDifferent*](#)
- [*distinctMembers*](#)

Restricted Cardinality:

- [*minCardinality*](#) (only 0 or 1)
- [*maxCardinality*](#) (only 0 or 1)
- [*cardinality*](#) (only 0 or 1)

Versioning:

- [*versionInfo*](#)
- [*priorVersion*](#)
- [*backwardCompatibleWith*](#)
- [*incompatibleWith*](#)
- [*DeprecatedClass*](#)
- [*DeprecatedProperty*](#)

Property Characteristics:

- [*ObjectProperty*](#)
- [*DatatypeProperty*](#)
- [*inverseOf*](#)
- [*TransitiveProperty*](#)
- [*SymmetricProperty*](#)
- [*FunctionalProperty*](#)
- [*InverseFunctionalProperty*](#)

Header Information:

- [*Ontology*](#)
- [*imports*](#)

Annotation Properties:

- [*rdfs:label*](#)
- [*rdfs:comment*](#)
- [*rdfs:seeAlso*](#)
- [*rdfs:isDefinedBy*](#)
- [*AnnotationProperty*](#)
- [*OntologyProperty*](#)

OWL DL and Full

2.2 OWL DL and Full Synopsis

The list of OWL DL and OWL Full language constructs that are in addition to or expand those of OWL Lite is given below.

Class Axioms:

- [*oneOf*](#) [*dataRange*](#)
- [*disjointWith*](#)
- [*equivalentClass*](#)
(applied to class expressions)
- [*rdfs:subClassOf*](#)
(applied to class expressions)

Arbitrary Cardinality:

- [*minCardinality*](#)
- [*maxCardinality*](#)
- [*cardinality*](#)

Boolean Combinations of Class Expressions:

- [*unionOf*](#)
- [*complementOf*](#)
- [*intersectionOf*](#)

Filler Information:

- [*hasValue*](#)

Sommaire

- Les classes OWL
 - Définition par subsomption
 - Définition par disjonction
- Les instances
- Les descriptions de classes
 - Par intersection
 - Par union
 - Par complément
 - Par énumération
- Les propriétés
 - Les propriétés objets
 - Les propriétés de données
 - Les caractéristiques des propriétés
- Les restrictions de propriétés
 - Les restrictions de valeurs
 - Les restrictions de cardinalités
- Les spécificités OWL2
- **Les règles**

Pour aller plus loin...

Les règles

- Les règles sont généralement écrites en SWRL
<https://www.w3.org/Submission/SWRL/>
- Elles sont composées de 2 parties principales :
un corps « body » et une tête « head ».

Les règles

- Ecrivons une règle permettant de dire qu'un animal qui mange un autre animal est un carnivore
- Puis créons l'animal `Alphonse` qui mange la girafe `Ernestine`

Les règles

The screenshot displays a software interface with two main panels. The left panel, titled 'Class hierarchy (inferred)', shows a tree structure of classes. The root is 'owl:Thing', which branches into 'Animal' and 'Plant'. 'Animal' further branches into 'Carnivore', 'Herbivore', 'Branch', 'Copines_d_Ernestine', 'Leaf', 'Les_girafes_copines', and 'Tree'. 'Carnivore' branches into 'Canide', 'Felide', and 'Omnivore'. The 'Animal' class is highlighted with a blue selection bar. The right panel, titled 'Rules', shows a rule editor. The rule text is 'Animal(?x), eats(?x, ?y), Animal(?y) -> Carnivore(?x)'. Below the rule editor, there are sections for 'Annotations: Animal' and 'Description: Animal'. The 'Description: Animal' section shows 'Equivalent To' and 'SubClass Of' options. The 'SubClass Of' option is selected, and the value is 'hasAge min 1 xsd:nonNegativeInteger'.

Interprétation logique

$\forall x, y \text{ Animal}(x) \wedge \text{Animal}(y) \wedge \text{eats}(x, y) \rightarrow \text{Carnivore}(x)$

Les règles

The screenshot displays a Semantic Web editor interface with several panels:

- Class hierarchy (inferred):** Shows a tree structure starting from `owl:Thing`, with `Animal` highlighted. Other classes include `Branch`, `Copines_d_Ernestine`, `Leaf`, `Les_girafes_copines`, and `Plant`.
- Individuals: Alphonse:** Lists individuals: `Alphonse`, `Basilic`, `Ernestine`, `Ernestine_la_girafe`, `Gertrude`, `Leon_le_lion`, and `Noemie`.
- Annotations: Alphonse:** Shows a list of annotations for the individual `Alphonse`.
- Description: Alphonse:** Shows the inferred types for `Alphonse`, including `Animal` and `Carnivore`. Both are circled in red.
- Property assertions: Alphonse:** Shows object property assertions for `Alphonse`, including `eats Ernestine` and `eats Ernestine_la_girafe`. Both are circled in red.

On infère de nouvelles connaissances :

- Alphonse est un carnivore

- Il mange aussi Ernestine_la_girafe par application du sameAs