

# Web Sémantique

Abir BENAZZOUZ – Grégoire HEBRAS

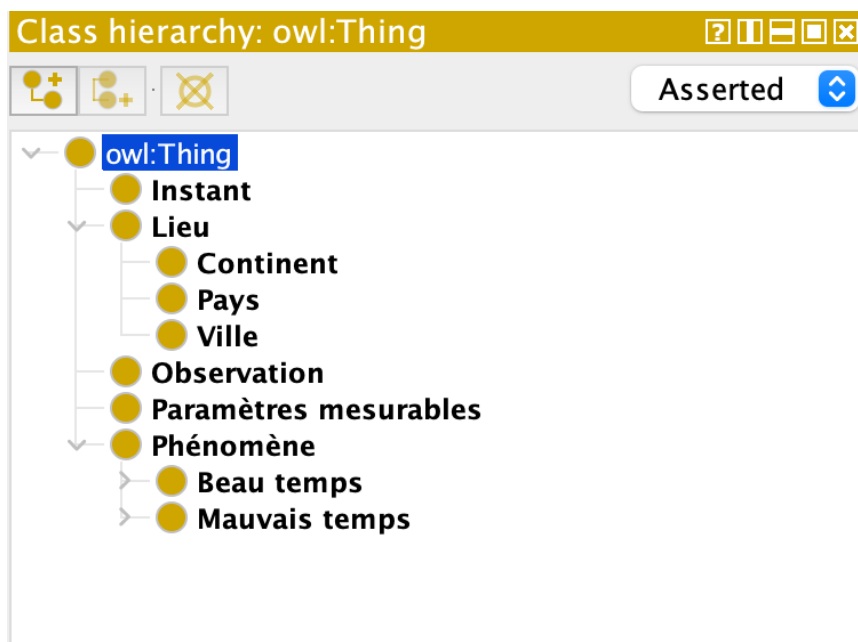
SISS

## TP 1 : Création d'une ontologie et utilisation du raisonneur dans Protégé

### 2.2.1 : Conception de l'ontologie légère

On commence par la conception de l'énoncé en créant les classes et sous-classes appropriées, à savoir les suivantes :

On note par exemple que « Beau Temps » et « Mauvais Temps » étant deux types de phénomènes, ils représentent des sous-classes de « Phénomène » dans notre ontologie.

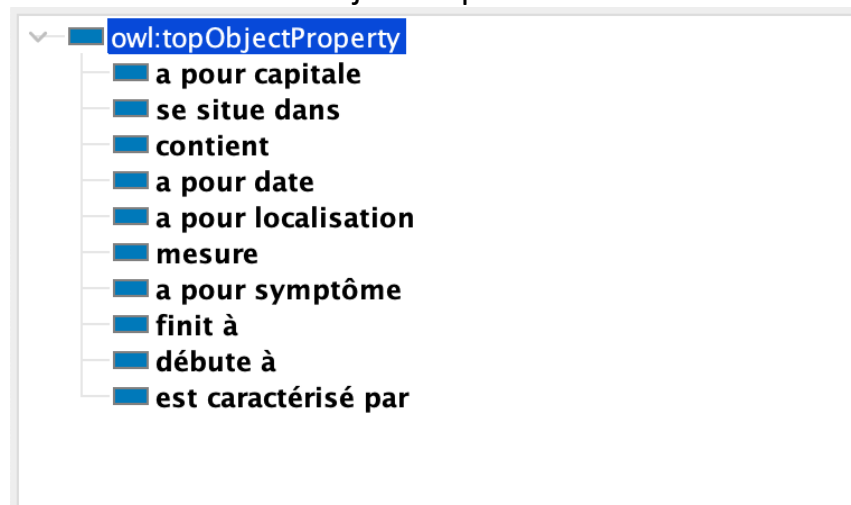


« Ensoleillement » est une sous-classe de « Beau Temps » et « Brouillard » et « Pluie » sont des sous-classes de « Mauvais Temps » :

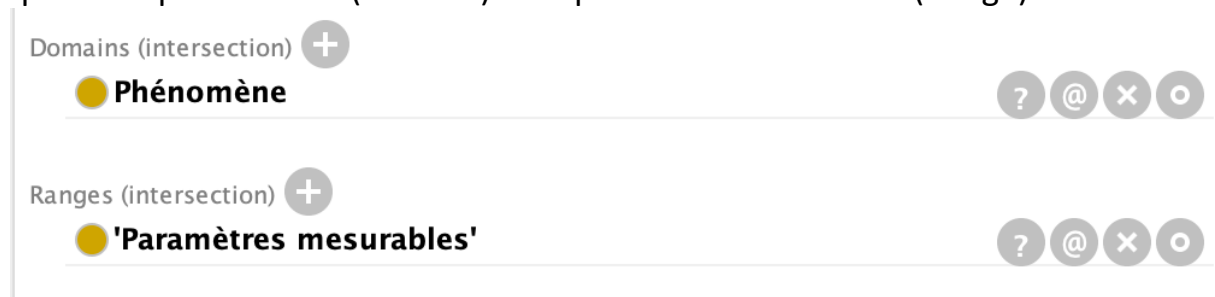


Toutes les classes sont elles-mêmes des sous-classes de owl :Thing

On crée ensuite les ObjectProperties suivantes :



Par exemple, on nous indique qu'un phénomène est caractérisé par des paramètres mesurables, on crée donc l'object property « est caractérisé par » qui lie un phénomène (Domain) à un paramètre mesurable (Range) :



Et enfin les DataProperties suivantes :



Par exemple, on nous indique qu'un phénomène a une durée en minutes, on crée donc la DataProperty « a une durée de » qui lie un phénomène (Domain) à une valeur de type integer (Range) :

The screenshot shows a web interface for configuring an ontology. It has two main sections: 'Domains (intersection)' and 'Ranges'. The 'Domains' section has a yellow circle icon and the text 'Phénomène'. The 'Ranges' section has a red circle icon and the text 'xsd:integer'. To the right of each section are four circular buttons: a question mark, an at-sign, a cross, and an empty circle.

| Section                | Value       | Buttons    |
|------------------------|-------------|------------|
| Domains (intersection) | Phénomène   | ?, @, x, o |
| Ranges                 | xsd:integer | ?, @, x, o |

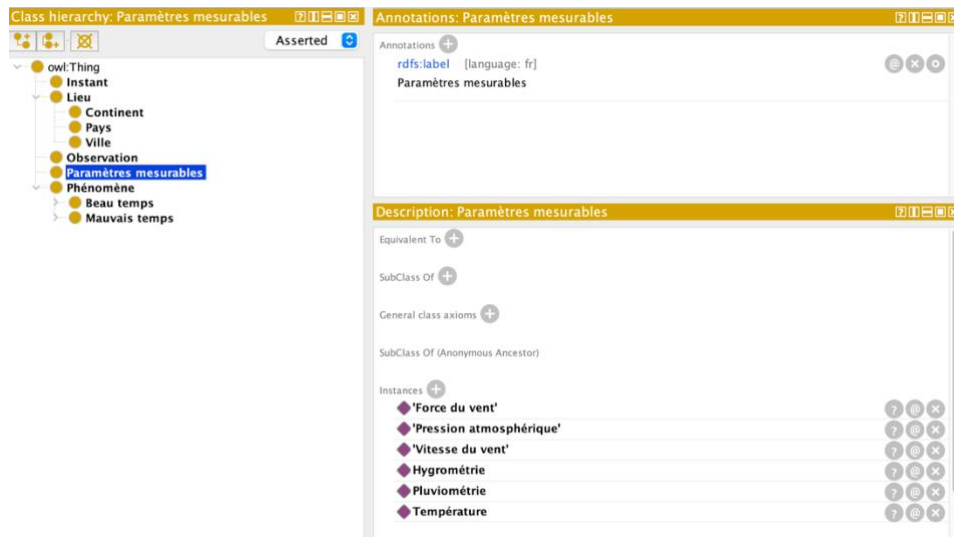
On utilise le même raisonnement pour créer toutes les autres classes, objectProperties et dataProperties, en prenant en compte les différents Domains et Ranges précisés

### 2.2.2 : Peuplement de l'ontologie légère

On lance le raisonneur après avoir représenté chacun des faits suivants et on observe ce qui se passe :

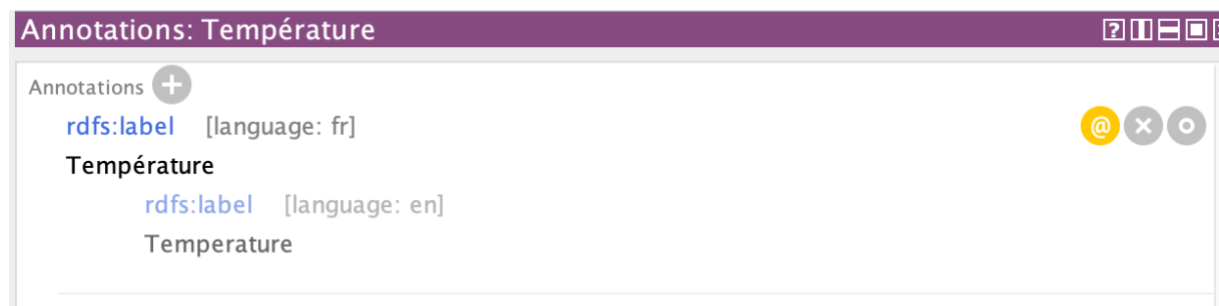
1. La température, l'hygrométrie, la pluviométrie, la pression atmosphérique, la vitesse du vent et la force du vent sont des paramètres mesurables (Attention, pas des types de paramètres, mais des instances de paramètres)

On constate déjà que les instances que l'on crée depuis l'onglet Individuals apparaissent déjà avant même de lancer le raisonneur dans la liste des instances de la classe paramètres mesurables



On lance alors le raisonneur, il ne se passe rien, il n'infère rien de particulier à ce stade

2. Le terme temperature est un synonyme anglais de température. Examinez les "Annotations" de l'individu.



On ajoute l'annotation Temperature comme synonyme en anglais

3. La force du vent est similaire à la vitesse du vent

On notifie que la force du vent est équivalente à la vitesse du vent et directement la force du vent apparaît également comme synonyme de la vitesse du vent.

Individuals: Force du vent

Force du vent

Hygrométrie

Pluviométrie

Pression atmosphérique

Température

Vitesse du vent

Annotations: Force du vent

Annotations

rdfs:label [language: fr]

Force du vent

Description: Force du vent

Types

'Paramètres mesurables'

Same Individual As

'Vitesse du vent'

Different Individuals

Individuals: Vitesse du vent

Force du vent

Hygrométrie

Pluviométrie

Pression atmosphérique

Température

Vitesse du vent

Annotations: Vitesse du vent

Annotations

rdfs:label [language: fr]

Vitesse du vent

Description: Vitesse du vent

Types

'Paramètres mesurables'

Same Individual As

'Force du vent'

Different Individuals

- Toulouse est située en France. Remarquez que les individus dans cette phrase ne sont pas typés : créez Toulouse et France non pas comme une ville et un pays, mais comme des individus sans classe. Comment les classifie le raisonneur ?

On crée deux individus Toulouse et France sans leur assigner de type. On précise juste grâce à une object property assertion que Toulouse ‘se situe dans’ France

The screenshot shows the Protégé interface with the following panels:

- Individuals: Toulouse**: A list of properties including Force du vent, France, Hygrométrie, Pluviométrie, Pression atmosphérique, Température, Toulouse (selected), and Vitesse du vent.
- Annotations: Toulouse**: Shows an annotation `rdfs:label` with the value `Toulouse` and a language of `fr`.
- Description: Toulouse**: Shows options for Types, Same Individual As, and Different Individuals.
- Property assertions: Toulouse**: Shows an object property assertion `'se situe dans' France`.

On lance le raisonneur, il est capable d’inférer seul que Toulouse et France sont de type lieu.

The screenshot shows the Protégé interface after running the reasoner. The inferred type `Lieu` is now visible in the **Description** panels for both individuals:

- Individuals: Toulouse**: The same list of properties as before.
- Annotations: Toulouse**: The same annotation as before.
- Description: Toulouse**: The **Types** section now includes `Lieu`.
- Property assertions: Toulouse**: The same object property assertion as before.
- Individuals: France**: The same list of properties as before.
- Annotations: France**: Shows an annotation `rdfs:label` with the value `France` and a language of `fr`.
- Description: France**: The **Types** section now includes `Lieu`.
- Property assertions: France**: Shows options for Object property assertions, Data property assertions, Negative object property assertions, and Negative data property assertions.

5. Toulouse est une ville  
Le raisonneur n'infère rien de nouveau.

6. La France a pour capitale Paris. Ici aussi, Paris est un individu non typé

On crée un individu Paris sans lui donner de type ni d'object property, mais on ajoute une object property assertion dans France pour préciser que Paris est la capitale de la France.

En lançant le raisonneur on constate qu'il est capable d'inférer que la France est un pays, et que Paris est une ville.

The screenshot displays a reasoning tool interface with four main panels arranged in a 2x2 grid. The top-left panel, titled 'Individuals: France', shows a list of individuals: Force du vent, France (selected), Hygrométrie, Paris, Pluviométrie, Pression atmosphérique, Température, Toulouse, and Vitesse du vent. The top-right panel, titled 'Annotations: France', shows an annotation for 'rdfs:label' with the value 'France' and a language of 'fr'. The bottom-left panel, titled 'Individuals: Paris', shows a list of individuals: Force du vent, France, Hygrométrie, Paris (selected), Pluviométrie, Pression atmosphérique, Température, Toulouse, and Vitesse du vent. The bottom-right panel, titled 'Annotations: Paris', shows an annotation for 'rdfs:label' with the value 'Paris' and a language of 'fr'. The middle-right panel, titled 'Property assertions: France', shows an object property assertion for 'a pour capitale' with the value 'Paris'. The middle-left panel, titled 'Description: France', shows the type 'Pays' (Country) inferred for France. The middle-right panel, titled 'Property assertions: Paris', shows no assertions. The middle-left panel, titled 'Description: Paris', shows the type 'Ville' (City) inferred for Paris.

7. Le 10/11/2015 `a 10h00 est un instant que l'on appellera I1 (not`e 2015-11-10T10 :00 :00Z)

Le raisonneur d`duit que I1 est un instant :

The screenshot displays a Semantic Web editor interface with four main panels:

- Individuals: I1**: A list of properties including Force du vent, France, Hygrom`trie, I1 (selected), Paris, Pluviom`trie, Pression atmosph`rique, Temp`rature, Toulouse, and Vitesse du vent.
- Annotations: I1**: Shows an annotation for `rdfs:label` with the value `I1` and a language of `fr`.
- Description: I1**: Shows the type `Instant` assigned to the individual.
- Property assertions: I1**: Shows a data property assertion for `'a pour timestamp'` with the value `"2015-11-10T10:00:00Z"^^`.

8. P1 est une observation qui a mesure la valeur 3 mm de pluviom`trie ` Toulouse ` l'instant I1 (pas besoin de repr`senter l'unit`)

The screenshot displays a Semantic Web editor interface with four main panels:

- Individuals: P1**: A list of properties including Force du vent, France, Hygrom`trie, I1, P1 (selected), Paris, Pluviom`trie, Pression atmosph`rique, Temp`rature, Toulouse, and Vitesse du vent.
- Annotations: P1**: Shows an annotation for `rdfs:label` with the value `P1` and a language of `fr`.
- Description: P1**: Shows the type `Observation` assigned to the individual.
- Property assertions: P1**: Shows a data property assertion for `'a pour valeur'` with the value `3`.

9. A1 a pour sympt`me P1



Individuals: P1

- Force du vent
- France
- Hygrométrie
- I1
- P1**
- Paris
- Pluviométrie
- Pression atmosphérique
- Température
- Toulouse
- Vitesse du vent

Annotations: P1

Annotations +

rdfs:label [language: fr]

P1

Description: P1

Types +

Same Individual As +

Different Individuals +

Property assertions: P1

Object property assertions +

'a pour date' I1

mesure Pluviométrie

Data property assertions +

'a pour valeur' 3

Negative object property assertions +

Negative data property assertions +

Description: P1

Types +

**Observation**

Same Individual As +

Different Individuals +

Property assertions: P1

Object property assertions +

'a pour date' I1

mesure Pluviométrie

Data property assertions +

'a pour valeur' 3

Negative object property assertions +

Le raisonneur déduit que P1 est une observation.

Description: A1

Types +

**Phénomène**

Same Individual As +

Different Individuals +

Property assertions: A1

Object property assertions +

'a pour symptôme' P1

Data property assertions +

Negative object property assertions +

Negative data property assertions +

Il déduit que A1 est un phénomène puisque l'object property "a pour symptôme" lie un phénomène à une observation.

Chara
Description: a pour symptôme

☐ Functional  
☐ Inverse function  
☐ Transitive  
☐ Symmetric  
☐ Asymmetric  
☐ Reflexive  
☐ Irreflexive

Equivalent To +  
SubProperty Of +  
Inverse Of +  
Domains (intersection) +  
● Phénomène  
Ranges (intersection) +  
● Observation  
Disjoint With +

### 2.2.3 : Ebauche de l'ontologie légère

Dans cette partie du TP, on exprime les connaissances suivantes :

1. Toute instance de ville ne peut pas être un pays

On disjoint Ville avec continent et pays :

Description: Ville

Equivalent To +  
SubClass Of +  
● Lieu  
General class axioms +  
SubClass Of (Anonymous Ancestor)  
Instances +  
◆ Toulouse  
Target for Key +  
Disjoint With +  
● Continent, Pays

La disjonction se fait ensuite automatiquement sur les classes « continent » et « pays ».

2. Un phénomène court est un phénomène dont la durée est de moins de 15 minutes
3. Un phénomène long est un phénomène dont la durée est au moins de 15 minutes

On ajoute ces expressions dans equivalent to :

The screenshot shows the Protégé interface with the 'Classes' tab selected. The 'Class hierarchy: Phénomène court' panel on the left shows a tree structure where 'Phénomène court' is highlighted. The 'Annotations: Phénomène court' panel on the right shows an annotation for 'rdfs:label' with the value 'Phénomène court'. The 'Description: Phénomène court' panel shows the 'Equivalent To' property set to 'Phénomène that 'a une durée de' some xsd:integer[< 15]'. The 'SubClass Of' property is set to 'Phénomène'. The 'Class expression editor' at the bottom displays the expression 'Phénomène that 'a une durée de' some xsd:integer[< 15]'.

4. Un phénomène long ne peut pas être un phénomène court

On disjoint phénomène long et phénomène court :

Description: Phénomène long

Equivalent To

Phénomène that 'a une durée de' some xsd:integer[>= 15]

SubClass Of

Phénomène

General class axioms

SubClass Of (Anonymous Ancestor)

Instances

Target for Key

Disjoint With

'Phénomène court'

- La propriété indiquant qu'un lieu est inclus dans un autre a pour propriété inverse la propriété indiquant qu'un lieu en inclue un autre.

On utilise la caractéristique inverse of :

Chara
Description: se situe dans

☐ Functional  
☐ Inverse function  
☐ Transitive  
☐ Symmetric  
☐ Asymmetric  
☐ Reflexive  
☐ Irreflexive

Equivalent To

SubProperty Of

Inverse Of

contient

Domains (intersection)

Lieu

Ranges (intersection)

Lieu

Disjoint With

SuperProperty Of (Chain)

6. Si un lieu A est situé dans un lieu B et que ce lieu B est situé dans un lieu C, alors le lieu A est situé dans le lieu C (utilisez les caractéristiques de la relation)

La propriété « se situe dans » est transitive :

Chara ? ? ? ? ? Description: se situe dans ? ? ? ? ?

☐ Functional

☐ Inverse functional

☒ Transitive

☐ Symmetric

☐ Asymmetric

☐ Reflexive

☐ Irreflexive

Equivalent To +

SubProperty Of +

Inverse Of +

**contient** ? @ x o

Domains (intersection) +

**Lieu** ? @ x o

Ranges (intersection) +

**Lieu** ? @ x o

7. A tout pays correspond une et une seule capitale (utilisez les caractéristiques de la relation).

On coche functional et inverse functional : Cette propriété ne peut alors être liée qu'à une seule valeur (il faut que la propriété soit bijective) :

Characteristics: a ? ? ? ? ? Description: a pour capitale ? ? ? ? ?

☒ Functional

☒ Inverse functional

☐ Transitive

☐ Symmetric

☐ Asymmetric

☐ Reflexive

☐ Irreflexive

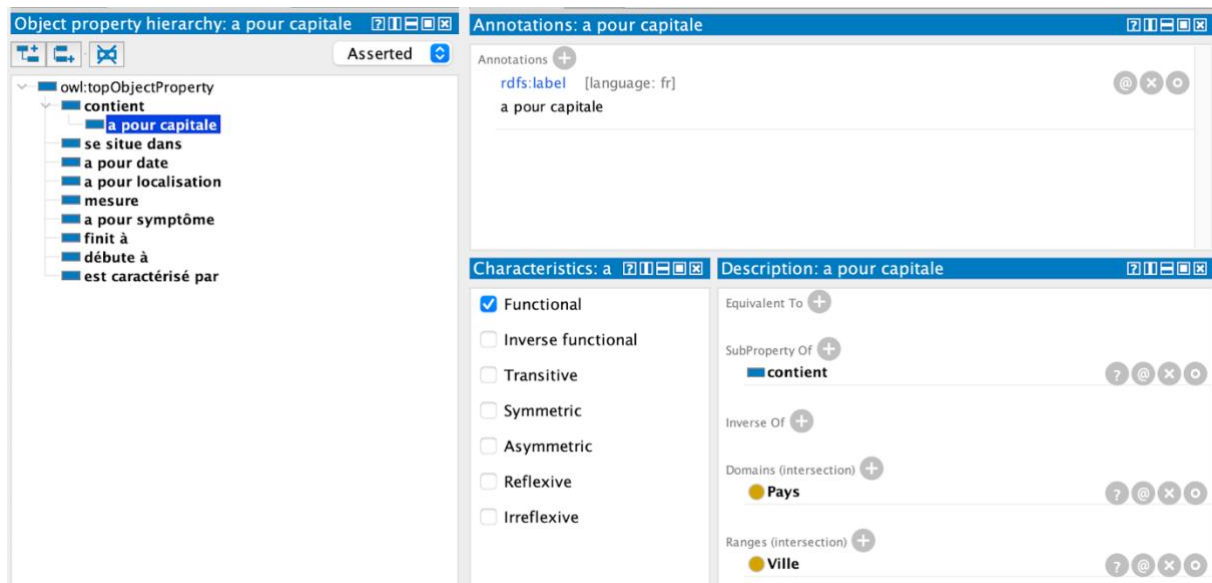
Equivalent To +

SubProperty Of +

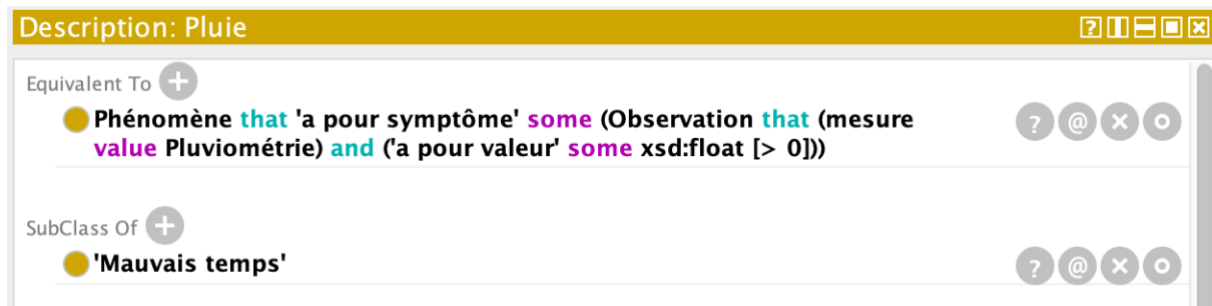
**contient** ? @ x o

8. Si un pays a pour capitale une ville, alors ce pays contient cette ville (utilisez la notion de sous-propriété).

« a pour capitale » devient une sous-propriété de contient :



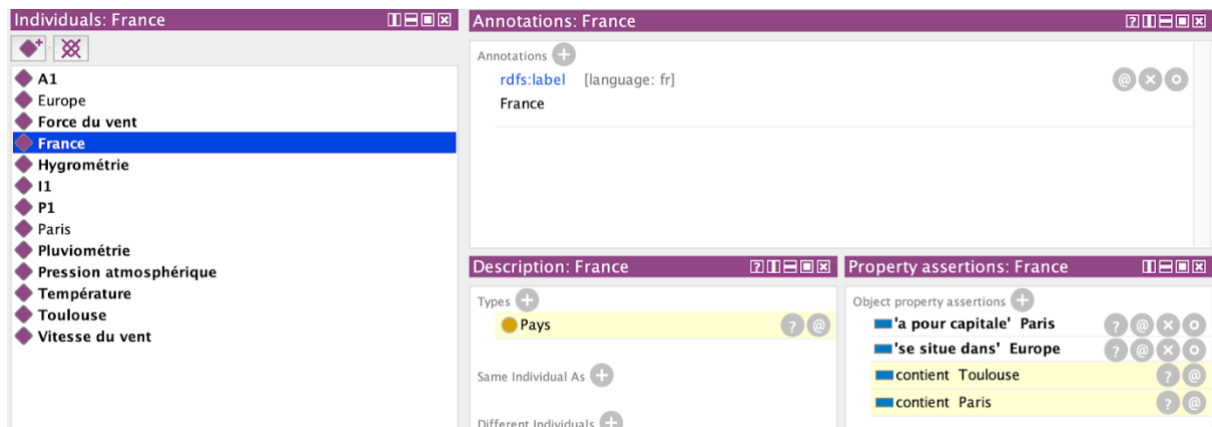
9. La Pluie est un phénomène ayant pour symptôme une Observation de Pluviométrie dont la valeur est supérieure à 0.



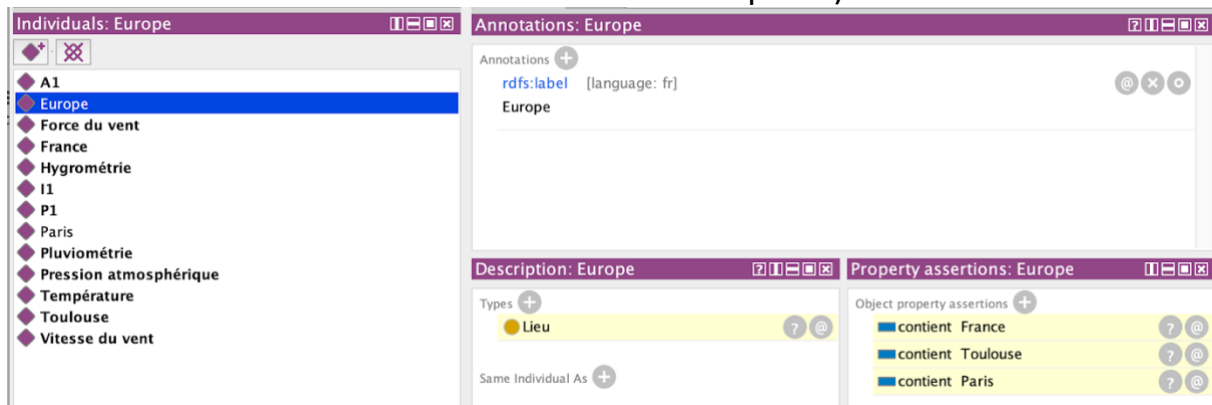
#### 2.2.4 :Peuplement de l'ontologie lourde

1. La France est située en Europe

Le raisonneur est maintenant capable d'inférer que la France contient Paris (puisque Paris est la capitale de la France et que “est la capitale de” est une sous-propriété de “contient”)



Au niveau de l'individu Europe, comme la propriété contient est transitive (comme elle est l'inverse de la propriété se situe dans qui a été définie comme transitive), le raisonneur infère à présent que Europe contient France (puisque'on a précisé que La France est située en Europe) mais que Europe contient également Toulouse (définie comme se situant dans France) et Paris (définie au niveau de l'individu France comme en étant la capitale)



## 2. Paris est la capitale de la France

On crée une data property “est la capitale de” qui lie une ville à un pays, est l'inverse de “a pour capitale” et est une sous-propriété de “se situe dans” et qui est “Inverse functional” et “Functional” puisque cette propriété est bijective :

| Characteristics: es ? 1 2 3 4 5  | Description: est la capitale de ? 1 2 3 4 5   |
|--|---|
| <input checked="" type="checkbox"/> Functional<br><input checked="" type="checkbox"/> Inverse functional<br><input type="checkbox"/> Transitive<br><input type="checkbox"/> Symmetric<br><input type="checkbox"/> Asymmetric<br><input type="checkbox"/> Reflexive<br><input type="checkbox"/> Irreflexive | Equivalent To +<br>SubProperty Of +<br><div> <span>'se situe dans'</span> <span>?</span> <span>@</span> <span>x</span> <span>o</span> </div> <div> <span>inverse (contient)</span> <span>?</span> <span>@</span> </div> Inverse Of +<br><div> <span>'a pour capitale'</span> <span>?</span> <span>@</span> <span>x</span> <span>o</span> </div> Domains (intersection) +<br><div> <span>Ville</span> <span>?</span> <span>@</span> <span>x</span> <span>o</span> </div> Ranges (intersection) +<br><div> <span>Pays</span> <span>?</span> <span>@</span> <span>x</span> <span>o</span> </div> Disjoint With + |

| Description: Paris ? 1 2 3 4 5   | Property assertions: Paris 1 2 3 4 5  |
|--|---|
| Types +<br><div> <span>Ville</span> <span>?</span> <span>@</span> </div> Same Individual As +<br>Different Individuals + | Object property assertions +<br><div> <span>'est la capitale de' France</span> <span>?</span> <span>@</span> </div> <div> <span>'se situe dans' France</span> <span>?</span> <span>@</span> </div> <div> <span>'se situe dans' Europe</span> <span>?</span> <span>@</span> </div> Data property assertions +<br>Negative object property assertions + |

Le raisonneur infère bien que Paris est la capitale de la France

### 3. La Ville Lumière est la capitale de la France

On crée l'individu "La ville lumière" et on lui donne comme propriété que c'est la capitale de la France



Description: La Ville Lumière

Types +

Ville ? @

Same Individual As +

Paris ? @

Different Individuals +

Property assertions: La Ville Lumière

Object property assertions +

'est la capitale de' France ? @ x o

'se situe dans' France ? @

'se situe dans' Europe ? @

Data property assertions +

Negative object property assertions +

Negative data property assertions +

Comme “est la capitale de” est inverse fonctionnel, le raisonneur déduit que la Ville lumière est équivalent à Paris

#### 4. Singapour est une ville et un pays

Le raisonneur n’accepte pas, puisque nous avons disjoint ville et pays

Description: Singapour

Types +

Pays ? @ x o

Ville ? @ x o

Same Individual As +

Different Individuals +

## OBSERVATIONS :

### 1. Que sait le raisonneur de A1 ?

Le raisonneur sait que A1 a pour symptôme P1 et infère que A1 est un phénomène puisque l'object property "a pour symptôme" lie un phénomène à une observation

The screenshot displays two panels from a reasoning tool. The left panel, titled "Description: A1", shows a list of types with "Phénomène" selected. Below this, there are buttons for "Same Individual As" and "Different Individuals". The right panel, titled "Property assertions: A1", shows a list of object property assertions with "'a pour symptôme' P1" selected. Below this, there are buttons for "Data property assertions", "Negative object property assertions", and "Negative data property assertions".

| Description: A1                   | Property assertions: A1   |
|-----------------------------------|---|
| <b>Types</b> +<br>● Phénomène ? @ | <b>Object property assertions</b> +<br>■ 'a pour symptôme' P1 ? @ × ○ |
| Same Individual As +              | Data property assertions +  |
| Different Individuals +           | Negative object property assertions +                                 |
|                                   | Negative data property assertions +                                   |

### 2. Que sait le raisonneur sur Paris ?

Le raisonneur sait que Paris est la capitale de la France, et en déduit donc que :

- Paris est une ville (puisque "est la capitale de" lie une ville à un pays)
- Paris est équivalent à La ville lumière (puisque La ville lumière a été définie comme étant la capitale de la France et que l'object property "est la capitale de" est bijective, le raisonneur déduit donc qu'il s'agit de la même entité)
- Paris se situe en France et en Europe (puisque "est la capitale de" est une sous-propriété de "se situe dans" et que "se situe dans" est une propriété transitive et que l'on a spécifié que la France se situe en Europe)

Description: Paris

Types

Ville

Same Individual As

'La Ville Lumière'

Different Individuals

Property assertions: Paris

Object property assertions

'est la capitale de' France

'se situe dans' France

'se situe dans' Europe

Data property assertions

Negative object property assertions

Negative data property assertions

- Constatez la réaction du raisonneur si Toulouse est déclarée comme la capitale de la France

Il s'agit là du même raisonnement que pour la Ville lumière, si on indique que Toulouse est la capitale de la France, comme la propriété "est la capitale de" est bijective, il en déduit alors qu'il s'agit du même individu que Paris et La Ville Lumière

Description: Toulouse

Types

Ville

Same Individual As

Paris, 'La Ville Lumière'

Different Individuals

Property assertions: Toulouse

Object property assertions

'se situe dans' France

'est la capitale de' France

'se situe dans' Europe

Data property assertions

Negative object property assertions

## TP 2 : Exploitation de l'ontologie

Une fois notre ontologie créée, nous allons implémenter lors de cette seconde partie des fonctions d'une API Java. Nous réutilisons pour cela l'ontologie créée sous Protégé au TP1, les noms des classes et propriétés sont différents car nous avons utilisé l'ontologie fournie afin d'être sûrs d'utiliser une ontologie correcte au cas où il ait des erreurs dans celle que nous avons créé au TP1.

Nous commençons par implémenter les fonctions de la classe `DoltYourselfModel.java` en vérifiant que l'implémentation est correcte grâce au test Junit associé :

- **createPlace()** : cette fonction crée une instance de la classe `Lieu`. Pour implémenter cette fonction, on récupère d'abord l'uri de la classe, qui est son identifiant unique grâce au label « Place ». La fonction `createPlace` ne prend qu'un seul argument : un `String`, le nom du lieu. On crée ensuite l'instance grâce à la fonction `createInstance()`

```
@Override
public String createPlace(String name) {
    List<String> uri = model.getEntityURI("Place");
    return model.createInstance(name, uri.get(0));
}
```

---

 testPlaceCreation [Runner: JUnit 4] (1,074 s)

- **createInstant()** : cette méthode permet d'instancier un instant, et de le lier à une donnée de timestamp à l'aide d'une data property. Cette fonction prend un seul argument également : un `TimestampEntity`.

```

@Override
public String createInstant(TimestampEntity instant) {
    //On stocke le timestamp
    String timestamp= instant.getTimeStamp();

    //On crée une instance d'instant et on récupère son uri
    String subject_uri = model.getEntityURI("Instant").get(0);
    String instance = model.createInstance("i"+timestamp, subject_uri);

    //On récupère l'uri de la data property timestamp
    String timestamp_uri = model.getEntityURI("has timestamp").get(0);

    //On ajoute la data property à l'instance
    model.addDataPropertyToIndividual(instance, timestamp_uri, timestamp);

    return instance;
}

```

---

 testInstantCreation [Runner: JUnit 4] (1,181 s)

- **getInstantURI()** : Cette fonction renvoie l'uri de l'instant donné en paramètre s'il existe et renvoie null sinon. Pour implémenter cette fonction on récupère tous les individus de la classe instant puis nous vérifions pour chacun s'ils sont liés à notre valeur de timestamp par la relation « has timestamp ». Si ce n'est le cas pour aucun, on renvoie null.

```

@Override
public String getInstantURI(TimestampEntity instant) {
    //On crée la variable résultat
    String resultat=null;

    // On récupère le timestamp
    String timestamp=instant.getTimeStamp();

    //On récupère la liste d'instances de la classe Instant
    List<String> instances_list = model.getInstancesURI( model.getEntityURI("Instant").get(0));

    //On récupère l'URI de la data property timestamp
    String timestamp_uri = model.getEntityURI("has timestamp").get(0);

    //Si
    for (String inst:instances_list) {
        if(model.hasDataPropertyValue(inst, timestamp_uri, timestamp)) {
            resultat=inst;
        }
    }
    return resultat;
}

```

---

 testInstantRetrieval [Runner: JUnit 4] (1,024 s)

- **getInstantTimestamp()** : Cette fonction renvoie le timestamp associé à l'instant passé en argument. Elle renvoie également null s'il n'est pas trouvé. Pour cela on récupère les couples propriété-objet associés à l'instant grâce à la méthode ListProperties. On regarde ensuite dans cette liste si une de ces propriétés est celle qui associe l'instant à un timestamp (« has timestamp ») si c'est le cas on renvoie le résultat, sinon on renvoie null.

```
@Override
public String getInstantTimestamp(String instantURI){
    //On crée la variable résultat
    String resultat=null;

    //On récupère les propriétés
    List<List<String>> listproperties = model.listProperties(instantURI);

    //On récupère l'URI de la data property timestamp
    String timestamp_uri = model.getEntityURI("has timestamp").get(0);

    //Si
    for (List<String> listproperty:listproperties) {
        if(listproperty.get(0).equals(timestamp_uri)) {
            resultat=listproperty.get(1);
        }
    }
    return resultat;
}
```

---

 testTimestampRetrieval [Runner: JUnit 4] (0,961 s)

- **createObs()** : Cette fonction crée une observation et renvoie l'uri de l'observation créée. Pour cela, on crée une instance d'observation, on associe à cette instance la valeur (grâce à l'uri de la propriété « has value »), puis on lui associe les paramètres (grâce à l'uri de la propriété « has characteristics »), puis on lui associe la date de la même manière. Enfin, on associe l'observation au sensor et on renvoie l'instance d'observation ainsi créée.

```

@Override
public String createObs(String value, String paramURI, String instantURI) {

    //Création du label observation avec timestamp intégré
    String label = "Obs__" + getInstantTimestamp(instantURI);

    //Création de l'instance
    String observation_uri = model.getEntityURI("Observation").get(0);
    String instance_uri = model.createInstance(label, observation_uri);

    //Association de l'instance à la valeur
    String hasValue_uri = model.getEntityURI("has value").get(0);
    model.addDataPropertyToIndividual(instance_uri , hasValue_uri ,value);

    //Association de l'instance aux paramètres
    String characteristics_uri = model.getEntityURI("has characteristics").get(0);
    model.addObjectPropertyToIndividual(instance_uri , characteristics_uri , paramURI);


    // Association de l'instance avec la data property timestamp
    String hasDate_uri = model.getEntityURI("has date").get(0);
    model.addObjectPropertyToIndividual(instance_uri , hasDate_uri ,instantURI);

    // Association de l'observation au sensor
    String sensor_uri = model.whichSensorDidIt(getInstantTimestamp(instantURI), paramURI);
    model.addObservationToSensor(instance_uri , sensor_uri);

    return instance_uri;
}

```

---

 testObservationCreation [Runner: JUnit 4] (1,092 s)

Tous les tests fonctionnent, y compris les test createSemanticModel et testParseTemperature :

---

 createSemanticModel [Runner: JUnit 4] (0,517 s)

---

 testParseTemperature [Runner: JUnit 4] (0,049 s)