

Ontologies et

Web Sémantique

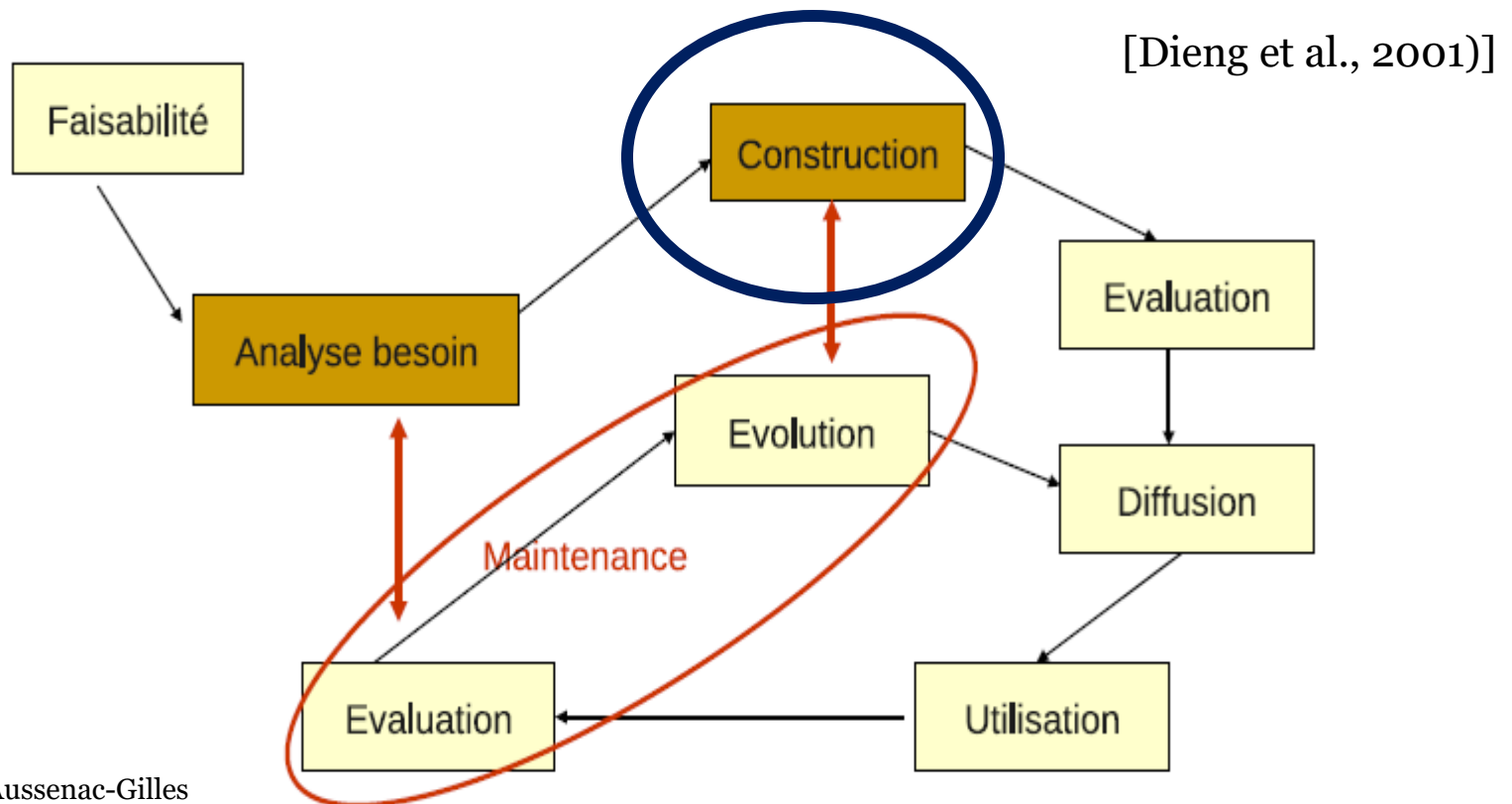
Les Ontologies – Formalisation et Exploitation

Plan du cours

1. Construction d'ontologies
2. Langages de formalisation et de représentation d'ontologies
3. Langages de développement d'ontologies
4. Quelques outils
5. OWL – Ontology Web Language

Cycle de vie d'une ontologie

- Les ontologies sont destinées à être utilisées comme des composants logiciels dans des systèmes répondant à des objectifs opérationnels différents.
- Leur développement doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel.



Cycle de vie d'une ontologie

➤ Construction d'une ontologie : 3 phases

- La **conceptualisation**: La conceptualisation d'un domaine nécessite l'identification de ses connaissances et le choix des entités à modéliser ainsi que leur organisation en une ontologie.
- L'**ontologisation**: formalisation, autant que possible, du modèle conceptuel obtenu à l'étape précédente.
- L'**opérationnalisation**: transcription de l'ontologie dans un langage formel et opérationnel de représentation de connaissances.
- ✓ L'étape d'ontologisation peut être complétée d'une étape *d'intégration* au cours de laquelle une ou plusieurs ontologies vont être importées dans l'ontologie à construire.

Langages de formalisation d'ontologies

- Langages de représentation de connaissances. Syntaxe et sémantique.
- Les travaux en Représentation des Connaissances ont donné naissance à plusieurs formalismes.
- Trois grandes familles :
 - Les Logiques de Description (DL)
 - Langage de Frames et les réseaux sémantiques
 - **Les graphes Conceptuels (CG)**

Langages de formalisation d'ontologies

- Langages de représentation de connaissances. Syntaxe et sémantique.
- Les travaux en Représentation des Connaissances ont donné naissance à plusieurs formalismes.
- Trois grandes familles :
 - Langage de Frames et les réseaux sémantiques :
 - Introduit par Minski.
 - Un frame représente soit une classe (class frame) ou un objet (instance frame).
 - Peut être vu comme un réseau de nœuds et relations.
 - Un frame contient des attributs (appelés slots) pour décrire les propriétés des objets.
 - Les valeurs des slots peuvent être spécifiées ou calculées.

Langages de formalisation d'ontologies

- Langages de représentation de connaissances. Syntaxe et sémantique.
- Les travaux en Représentation des Connaissances ont donné naissance à plusieurs formalismes.
- Trois grandes familles :
 - Les Logiques de Description (DL) :
- Donnent une sémantique solide aux langages de frame.
- Représentent les connaissances relatives à un domaine de référence à l'aide de "descriptions" qui peuvent être des concepts (classes), des rôles (relations) et des individus (objets).

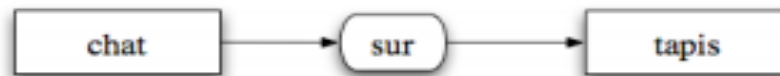
| | | |
|-------------|---|---|
| Woman | ≡ | Person \sqcap Female |
| Man | ≡ | Person \sqcap \neg Woman |
| Mother | ≡ | Woman \sqcap \exists hasChild.Person |
| Father | ≡ | Man \sqcap \exists hasChild.Person |
| Parent | ≡ | Father \sqcup Mother |
| Grandmother | ≡ | Mother \sqcap \exists hasChild.Parent |

Langages de formalisation d'ontologies

- Langages de représentation de connaissances. Syntaxe et sémantique.
- Les travaux en Représentation des Connaissances ont donné naissance à plusieurs formalismes.
- Trois grandes familles :
 - Les graphes Conceptuels (CG) :
- Notation graphique pour la logique.

« le chat est sur le tapis »

Forme graphique (Display Form – DF) :



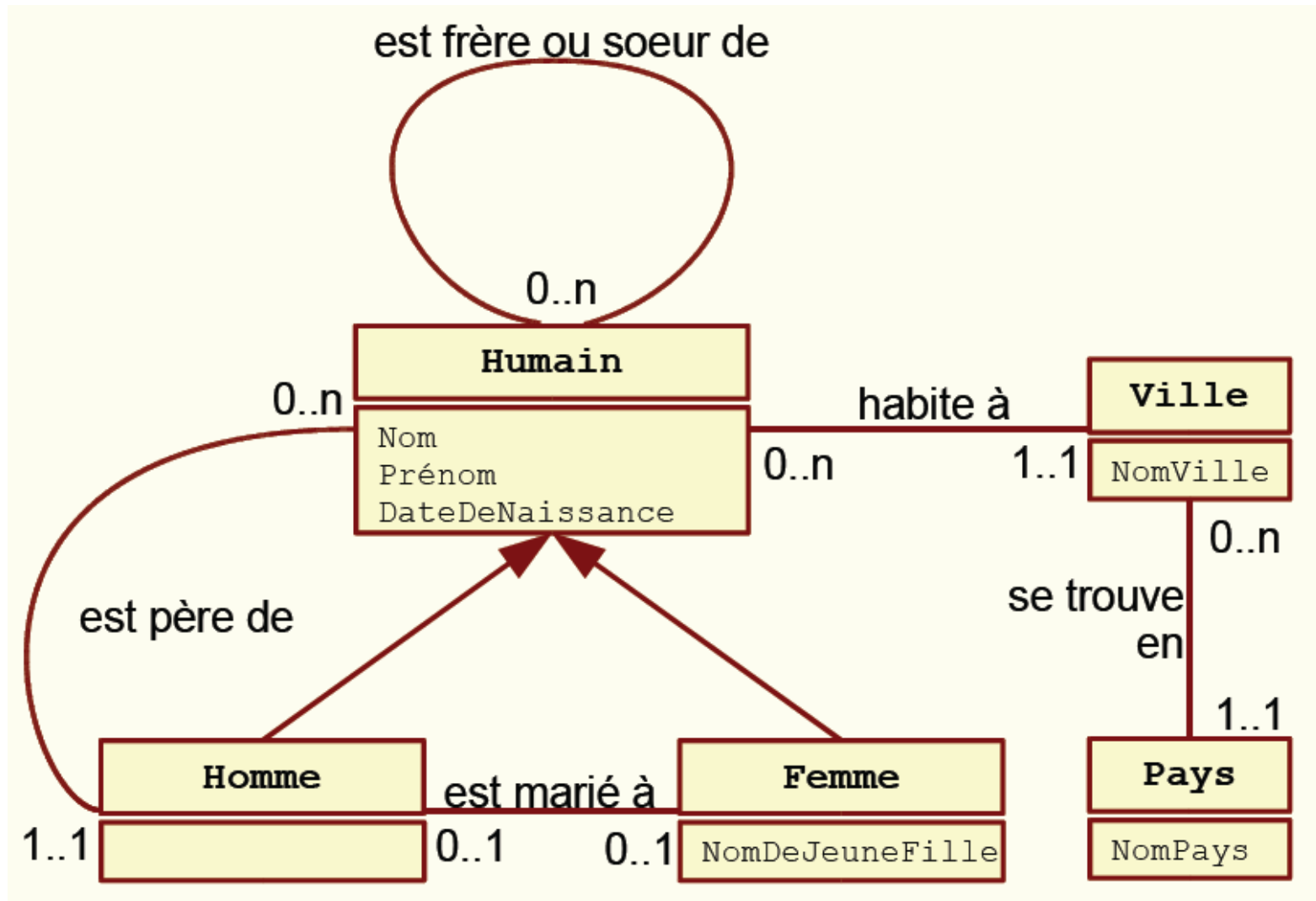
2 concepts : chat et tapis et 1 relation : sur

Forme linéaire (Linear Form – LF) :

[chat]->(sur)->[tapis]

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL - Exemple

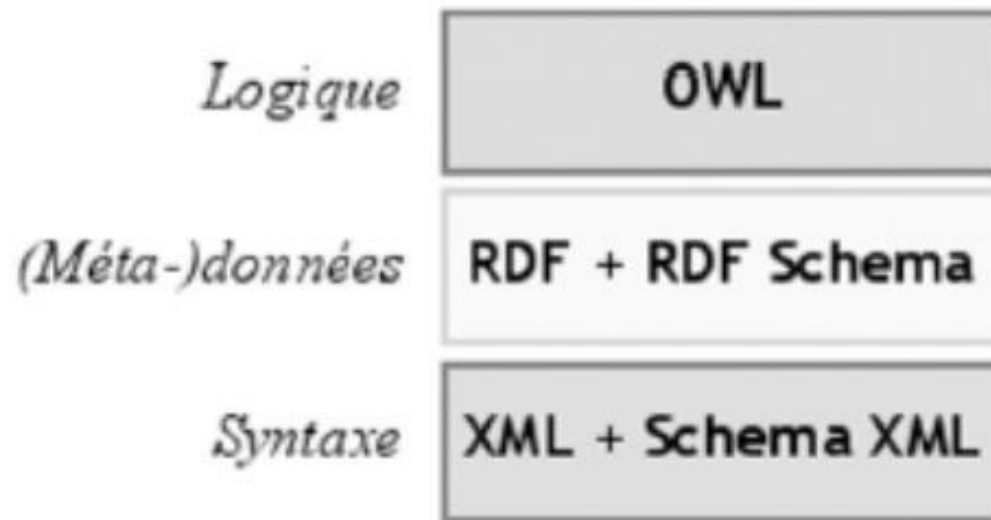


Langages de développement d'ontologies

- Description. Développement. Exploitation. Définition. Etc.
- **OWL** (Ontology Web Language), est un langage de **description** et de **développement** d'ontologies conçu pour la **publication** et le **partage** d'ontologies sur le **Web sémantique**.
- OWL : Fusion entre OIL (Ontology Inference Layer) et DAML (DARPA Agent Markup).
- Conçu pour ajouter **plus de sémantique et d'expressivité** à RDF et RDFS ainsi que des mécanismes d'**inférence**.
- OWL intègre des outils de comparaison des propriétés et des classes : identité, équivalence, contraire, cardinalité, symétrie, transitivité, etc.
- Basé **XML**. Fondé sur la syntaxe RDF.
- Document OWL. Extension **.owl** ou **.rdf**. Namespace : <http://www.w3.org/2002/07/owl#>.

Langages de développement d'ontologies

- Description. Développement. Exploitation. Définition. Etc.
- **OWL** (Ontology Web Language), est un langage de **description** et de **développement** d'ontologies conçu pour la **publication** et le **partage** d'ontologies sur le **Web sémantique**.



OWL - Ontology Web Language

➤ Différentes déclinaisons/profils de OWL :

1. OWL Lite :

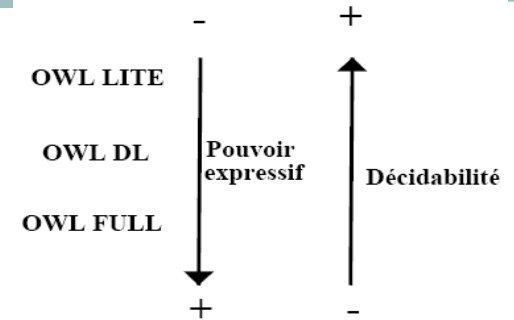
- Destiné aux utilisateurs qui ont besoin d'une hiérarchie de concepts simple.
- Utile aux applications qui ont besoin des hiérarchies de classifications et des caractéristiques de contraintes simples.

2. OWL DL :

- Fondé sur la logique descriptive.
- Utile aux applications qui demandent un maximum d'expressivité tout en garantissant la complétude et la décidabilité.
- Contient tous les constructeurs de OWL mais avec des restrictions.

3. OWL FULL :

- Permet le plus haut niveau d'expressivité.
- Destiné aux applications où il est plus important d'avoir un haut niveau de capacité de description, sans garantir la complétude et la décidabilité des calculs liés à l'ontologie.



OWL - Ontology Web Language

➤ Structure d'une ontologie OWL

OWL Lite

- RDF Schema Features:
 - [Class](#)
 - [rdf:Property](#)
 - [rdfs:subClassOf](#)
 - [rdfs:subPropertyOf](#)
 - [rdfs:domain](#)
 - [rdfs:range](#)
 - [Individual](#)
- (In)Equality:
 - [equivalentClass](#)
 - [equivalentProperty](#)
 - [sameAs](#)
 - [differentFrom](#)
 - [allDifferent](#)
- Property Characteristics:
 - [inverseOf](#)
 - [TransitiveProperty](#)
 - [SymmetricProperty](#)
 - [FunctionalProperty](#)
 - [InverseFunctionalProperty](#)
- Property Type Restrictions:
 - [allValuesFrom](#)
 - [someValuesFrom](#)
- Restricted Cardinality:
 - [minCardinality](#) (only 0 or 1)
 - [maxCardinality](#) (only 0 or 1)
 - [cardinality](#) (only 0 or 1)
- Header Information:
 - [ontology](#)
 - [imports](#)
- Class Intersection:
 - [intersectionOf](#)
- Versioning:
 - [versionInfo](#)
 - [priorVersion](#)
 - [backwardCompatibleWith](#)
 - [incompatibleWith](#)
 - [DeprecatedClass](#)
 - [DeprecatedProperty](#)
- Annotation Properties:
 - [rdfs:label](#)
 - [rdfs:comment](#)
 - [rdfs:seeAlso](#)
 - [rdfs:isDefinedBy](#)
- Datatypes
 - [DatatypeProperty](#)

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL

OWL DL and FULL

- Class Axioms:
 - [oneOf](#), [dataRange](#)
 - [disjointWith](#)
 - [equivalentClass](#)
(applied to class expressions)
 - [rdfs:subClassOf](#)
(applied to class expressions)
- Boolean Combinations of Class Expressions:
 - [unionOf](#)
 - [intersectionOf](#)
 - [complementOf](#)
- Arbitrary Cardinality:
 - [minCardinality](#)
 - [maxCardinality](#)
 - [cardinality](#)
- Filler Information:
 - [hasValue](#)

Quelques restrictions pour OWL DL par rapport à OWL FULL

OWL - Ontology Web Language

➤ **Structure d'une ontologie OWL**

- OWL est **ouvert**.
- Possibilité d'étendre des ontologies existantes .
- Employer diverses ontologies existantes pour compléter la définition d'une nouvelle ontologie.
- **Tout document OWL est une ontologie :**
 - ✓ qui peut avoir un **identificateur unique** représenté par une **URI**
 - ✓ qui contient :
 - des **faits** qui sont des descriptions d'individus
 - des **axiomes** qui fournissent les descriptions de concepts

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL

Différentes syntaxes de OWL : Il y a différentes syntaxes pour stocker, partager, éditer des ontologies OWL :

- La syntaxe d'échange **RDF/XML** officiellement recommandée, et que tout outil compatible OWL doit prendre en charge.
- Des **syntaxes standard** que tous les outils OWL prennent en charge.
- Des **syntaxes spécialement conçues** pour des applications et buts particuliers.
- Quelle que soit la syntaxe utilisée, le **langage OWL** n'est pas défini à l'aide d'une **syntaxe concrète particulière**, mais est défini par une **spécification structurelle abstraite** de haut niveau, qui est ensuite **traduite** dans diverses **syntaxes concrètes** :
 - ✓ Fonctionnelle, **RDF/XML**, Turtle, OWL/XML, Manchester.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

1. **Espaces de nommage** : indiquer de quels vocabulaires les termes de l'ontologie proviennent.

```
<rdf:RDF
  xmlns = "http://domain.tld/path/humanite#"
  xmlns:humanite= "http://domain.tld/path/humanite#"
  xml:base = "http://domain.tld/path/humanite"
  xmlns:vivant = "http://otherdomain.tld/otherpath/vivant#"
  xmlns:owl = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema#">
  ...
</rdf:RDF>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

1. Espaces de nommage : Abréviation

```
<!DOCTYPE rdf:RDF [  
    <!ENTITY humanite "http://domain.tld/path/humanite#" >  
    <!ENTITY vivant "http://otherdomain.tld/otherpath/vivant#" >  

```

```
<rdf:RDF  
    xmlns = "&humanite;"  
    xmlns:humanite= "&humanite;"  
    xml:base = "&humanite;"  
    xmlns:vivant = "&vivant;"  
    xmlns:owl = "http://www.w3.org/2002/07/owl#"  
    xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
    xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"  
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema#">
```

...

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

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

2. Entêtes d'une ontologie : décrit le contenu de l'ontologie

```
<owl:Ontology rdf:about="">  
  <rdfs:comment>Ontologie décrivant l'humanité</rdfs:comment>  
  <owl:imports  
    rdf:resource="http://otherdomain.tld/otherpath/vivant"/>  
  <rdfs:label>Ontologie sur l'humanité</rdfs:label>  
</owl:Ontology>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Il existe dans toute ontologie OWL une superclasse, nommée **Thing**, dont toutes les autres classes sont des sous-classes. **owl:Thing**
- **Description de classe** : type 1

```
<owl:Class rdf:ID="Humain" />  
<owl:Class rdf:ID="Femme" />  
<owl:Class rdf:ID="Homme" />  
<owl:Class rdf:ID="Ville" />  
<owl:Class rdf:ID="Pays" />
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Description de classe : Axiomes - Héritage

```
<owl:Class rdf:ID="Humain">
    <rdfs:subClassOf rdf:resource="#vivant;#EtreVivant" />
</owl:Class>

<owl:Class rdf:ID="Homme">
    <rdfs:subClassOf rdf:resource="#Humain" />
</owl:Class>

<owl:Class rdf:ID="Femme">
    <rdfs:subClassOf rdf:resource="#Humain" />
</owl:Class>
```

Le sujet d'une déclaration **rdfs:subClassOf** doit être un ID de classe, et l'objet doit être un ID de classe ou bien une restriction de propriété.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Description de classe : Axiomes - Equivalence : la classe équivalente a exactement la même extension (individus).

```
<owl:Class rdf:ID="Humain">
    <rdfs:subClassOf rdf:resource="&vivant;#EtreVivant" />
</owl:Class>

<owl:Class rdf:ID="Homme">
    <rdfs:subClassOf rdf:resource="#Humain" />
    <owl:equivalentClass rdf:resource="&vivant;#Male" />
</owl:Class>
```

Le sujet d'une déclaration **owl:equivalentClass** doit être un ID de classe, et l'objet doit être un ID de classe ou bien une restriction de propriété.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Description de classe : Axiomes - Disjonction: deux classes disjointes n'ont aucun membre commun dans leurs extensions.

```
<owl:Class rdf:ID="Humain">
    <rdfs:subClassOf rdf:resource="&vivant;#EtreVivant" />
</owl:Class>

<owl:Class rdf:ID="Homme">
    <rdfs:subClassOf rdf:resource="#Humain" />
    <owl:disjointWith rdf:resource="#Femme" />
</owl:Class>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Description de classe : Complément (Opérateur NON)

```
<owl:Class rdf:ID="NonHumain">  
    <owl:complementOf rdf:resource="#Humain" />  
</owl:Class>
```

La classe **NonHumain** inclut tous les individus du discours qui ne sont pas **Humain**.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Description de classe : Intersection (Opérateur ET)

```
<owl:Class rdf:ID="AfricanCapital">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#AfricanCity" />
    <owl:Class rdf:about="#Capital" />
  </owl:intersectionOf>
</owl:Class>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Description de classe : Union (Opérateur OU)

```
<owl:Class rdf:ID="Humain">  
  <owl:unionOf rdf:parseType="Collection">  
    <owl:Class rdf:about="#Femme" />  
    <owl:Class rdf:about="#Homme" />  
  </owl:unionOf>  
</owl:Class>
```

La classe **Humain** inclut les individus des classes **Femme** et **Homme**.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- **Description de classe par énumération** des individus d'une classe : Définir une classe en énumérant tous ses membres.

```
<owl:Class rdf:ID="Ville">  
  <owl:oneOf rdf:parseType="Collection">  
    <owl:Thing rdf:about="#Alger" />  
    <owl:Thing rdf:about="#Oran" />  
    <owl:Thing rdf:about="#Annaba" />  
  </owl:oneOf>  
</owl:Class>
```

Relative ID

OWL - Ontology Web Language

➤ **Structure d'une ontologie OWL – en syntaxe **RDF/XML****

3. Corps d'une ontologie : **Les Concepts / Classes**

- Description de classe par restriction de propriété
- Une restriction de propriété est un type particulier de description de classe.
- Elle décrit une classe anonyme, c'est-à-dire la classe de tous les individus satisfaisant à la restriction.
- Le langage OWL distingue deux types de restrictions de propriété : celles contraignant sa valeur et celles contraignant sa cardinalité.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Restrictions : contrainte de valeur et contrainte de cardinalité.
- Une contrainte de cardinalité porte sur le nombre de valeurs que peut prendre une propriété. *cardinality, minCardinality, maxCardinality*.

```
<owl:Class rdf:ID="Humain">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#aPourPere"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
        1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Restrictions : contrainte de valeur et contrainte de cardinalité.
- Une contrainte de valeur s'exerce sur la valeur d'une certaine propriété de l'individu. *hasValue*, *allValuesFrom*, *someValuesFrom*.

```
<owl:Class rdf:ID="Homme">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#genre" />
      <owl:hasValue rdf:resource="#Masculin" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Restrictions : contrainte de valeur et contrainte de cardinalité.
- Une contrainte de valeur s'exerce sur la valeur d'une certaine propriété de l'individu.

```
<owl:Class rdf:ID="Ville">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#seTrouveA" />
      <owl:allValuesFrom rdf:resource="#Pays" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Pour toutes les villes, si elles ont une localisation, toutes les localisations sont des Pays.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Restrictions : contrainte de valeur et contrainte de cardinalité.
- Une contrainte de valeur s'exerce sur la valeur d'une certaine propriété de l'individu.

```
<owl:Class rdf:ID="Ville">  
  <rdfs:subClassOf>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#seTrouveA" />  
      <owl:someValuesFrom rdf:resource="#Pays" />  
    </owl:Restriction>  
  </rdfs:subClassOf>  
</owl:Class>
```

*Pour toutes les villes, si elles ont une localisation, elles ont **au moins** une localisations de type Pays.*

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

```
<owl:Class rdf:ID="AlgiersThings">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#seTrouveA" />
      <owl:someValuesFrom rdf:resource="#Alger" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:ID="AlgiersThings">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#seTrouveA" />
      <owl:someValuesFrom rdf:resource="#Alger" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- La différence entre ces deux écritures :
 - **`rdfs:subClassOf`** :
 - Les ressources qui se trouvent à Alger ne sont pas nécessairement des AlgiersThings.
 - Exprime une condition nécessaire.
 - **`owl:equivalentClass`** :
 - Si une ressources se trouve à Alger, alors elle doit être dans la classe AlgiersThings.
 - Exprime une condition nécessaire et suffisante.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Description de classe : Intersection (Opérateur ET)

```
<owl:Class rdf:ID="AfricanCity">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Ville" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#locatedIn" />
      <owl:hasValue rdf:resource="#Afrique" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Si une owl:Thing est une **Ville** et se trouve en **Afrique** alors cette dernière est une instance de **AfricanCity**.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

3. Corps d'une ontologie : **Les Concepts / Classes**

- Description de classe : Complément

```
<owl:Class rdf:ID="NonAfricanCity">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Ville"/>
    <owl:Class>
      <owl:complementOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#locatedIn" />
          <owl:hasValue rdf:resource="#Africa" />
        </owl:Restriction>
      </owl:complementOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

NonAfricanCity inclut toutes les villes qui ne se trouvent pas en **Africa**.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

4. Corps d'une ontologie : **Les Propriétés**

- Deux types :
 - Les propriétés d'objet – **owl:ObjectProperty** : relier des instances à d'autres instances.

```
<owl:ObjectProperty rdf:ID="habite">  
  <rdfs:domain rdf:resource="#Humain" />  
  <rdfs:range rdf:resource="#Pays" />  
</owl:ObjectProperty>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

4. Corps d'une ontologie : **Les Propriétés**

- Deux types :
 - Les propriétés d'objet – **owl:ObjectProperty** : relier des instances à d'autres instances.
 - Les propriétés de type de donnée – **owl:DatatypeProperty** : relier des individus à des valeurs (RDF literals et XML Schema datatypes).

```
<owl:DatatypeProperty rdf:ID="age">  
  <rdfs:domain rdf:resource="#Humain" />  
  <rdfs:range rdf:resource="&xsd;positiveInteger"/>  
</owl:DatatypeProperty>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

4. Corps d'une ontologie : **Les Propriétés**

■ Héritage :

```
<owl:ObjectProperty rdf:ID="estDeLaFamilleDe">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="#Humain" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="aPourFrere">
  <rdfs:subPropertyOf rdf:resource="#estDeLaFamilleDe" />
  <rdfs:range rdf:resource="#Humain" />
...
</owl:ObjectProperty>
```

➔ toute entité ayant une propriété *aPourFrere* d'une certaine valeur a aussi une propriété *estDeLaFamilleDe* de même valeur.

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

4. Corps d'une ontologie : **Les Propriétés**

- Toutes les classes ont une propriété *locatedIn* vers Region :

```
<owl:ObjectProperty rdf:ID="locatedIn">  
  <rdfs:domain rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>  
  <rdfs:range rdf:resource="#Region" />  
</owl:ObjectProperty>
```


OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

4. Corps d'une ontologie : **Les Propriétés**

- Caractéristiques : transitivité (&owl;TransitiveProperty), la **symétrie** (&owl;SymmetricProperty), l'inverse (&owl;inverseOf), etc.

```
<owl:ObjectProperty rdf:ID="estMarieA">  
  <rdf:type rdf:resource="&owl;SymmetricProperty" />  
  <rdfs:domain rdf:resource="#Humain" />  
  <rdfs:range rdf:resource="#Humain" />  
</owl:ObjectProperty>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

4. Corps d'une ontologie : **Les Propriétés**

- Caractéristiques : transitivité (&owl;TransitiveProperty), la symétrie (&owl;SymmetricProperty), **l'inverse** (&owl;inverseOf), etc.

```
<owl:ObjectProperty rdf:ID="aPourEnfant">  
  <owl:inverseOf rdf:resource="#aPourParent" />  
  <rdfs:domain rdf:resource="#Humain" />  
  <rdfs:range rdf:resource="#Humain" />  
</owl:ObjectProperty>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

4. Corps d'une ontologie : **Les Propriétés**

- Caractéristiques : transitivité (&owl;TransitiveProperty), la symétrie (&owl;SymmetricProperty), l'inverse (&owl;inverseOf), etc.

```
<owl:ObjectProperty rdf:ID="aPourFemme">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Homme" />
  <rdfs:range rdf:resource="#Femme" />
</owl:ObjectProperty>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

5. Corps d'une ontologie : **Les Individus**

```
<owl:NamedIndividual rdf:ID="Stiegler">
  <rdf:type rdf:resource="#Homme">
  <nom>Stiegler</nom>
  <prenom>Bernard</prenom>
  <dateDeNaissance rdf:datatype="&xsd:date">
    1952-04-01
  </dateDeNaissance>
  <aUnLienDeFraternite rdf:resource="#XXXX" />
  <aPourPere rdf:resource="#YYYY" />
  <habiteA rdf:resource="#Paris" />
</owl:NamedIndividual >
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

5. Corps d'une ontologie : **Les Individus**

```
<Homme rdf:ID="Stiegler">
  <nom>Stiegler</nom>
  <prenom>Bernard</prenom>
  <dateDeNaissance rdf:datatype="&xsd:date">
    1952-04-01
  </dateDeNaissance>
  <aUnLienDeFraternite rdf:resource="#XXXX" />
  <aPourPere rdf:resource="#YYYY" />
  <habiteA rdf:resource="#Paris" />
</Homme>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

5. Corps d'une ontologie : **Les Individus**

- rdf:type est la propriété RDF qui lie un individu à une classe dont il est membre.
- Ces deux écritures sont possibles :

```
<owl:NamedIndividual rdf:ID="Stiegler">  
    <rdf:type rdf:resource="#Homme"/>  
</owl:NamedIndividual>
```

```
<owl:Thing rdf:ID="Stiegler">  
    <rdf:type rdf:resource="#Homme"/>  
</owl:Thing>
```

OWL - Ontology Web Language

➤ **Structure d'une ontologie OWL – en syntaxe **RDF/XML****

5. Corps d'une ontologie : **Les Individus**

- Indiquer que deux individus sont identiques :

```
<Ville rdf:ID="Alger" />
```

```
<Ville rdf:ID="ElBahdja">
```

```
    <owl:sameAs rdf:resource="#Alger"/>
```

```
</Ville>
```

OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

5. Corps d'une ontologie : **Les Individus**

- Indiquer que deux individus sont différents :

```
<Couleur rdf:ID="Bleu" />
```

```
<Couleur rdf:ID="Rouge">
```

```
  <owl:differentFrom rdf:resource="#Bleu"/>
```

```
</Couleur>
```

```
<Couleur rdf:ID="Jaune">
```

```
  <owl:differentFrom rdf:resource="#Bleu"/>
```

```
  <owl:differentFrom rdf:resource="#Rouge"/>
```

```
</Couleur>
```


OWL - Ontology Web Language

➤ Structure d'une ontologie OWL – en syntaxe **RDF/XML**

5. Corps d'une ontologie : **Les Individus**

- Indiquer que deux individus sont différents :

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <design:Couleur rdf:about="#Bleu" />
    < design:Couleur rdf:about="#Rouge" />
    < design:Couleur rdf:about="#Jaune" />
  </owl:distinctMembers>
</owl:AllDifferent>
```

Quelques outils

Protégé: IDE - édition, visualisation, contrôle d'ontologie, l'extraction d'ontologies à partir de sources textuelles, et la fusion semi-automatique d'ontologies. <https://protege.stanford.edu/>

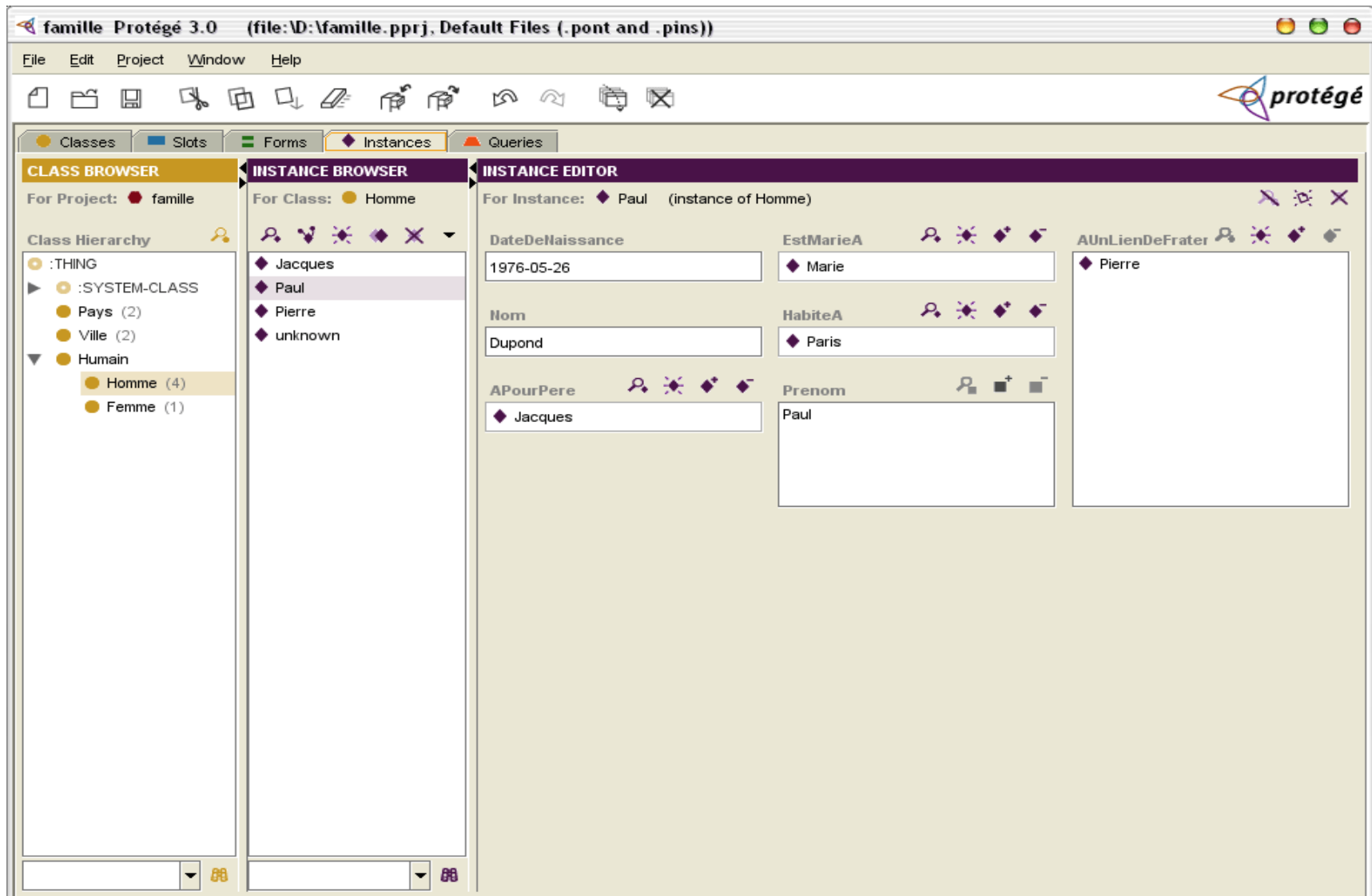
The screenshot displays the Protégé 3.0 IDE interface. The title bar indicates the file is 'famille.pprj'. The menu bar includes File, Edit, Project, Window, and Help. The toolbar contains various icons for file operations and editing. The main interface is divided into several panes:

- CLASS BROWSER**: Shows the class hierarchy for the project 'famille'. The hierarchy is: :THING (parent), :SYSTEM-CLASS (child), Pays (child), Ville (child), Humain (child), and Homme (child of Humain). The 'Homme' class is selected.
- CLASS EDITOR**: Shows the details for the 'Homme' class. It includes fields for Name (Homme), Role (Concrete), and Documentation. There are also buttons for Constraints and a Template Slots section.
- Template Slots**: A table listing the slots for the 'Homme' class.

| Name | Cardinality | Type | Other Facets |
|---------------------|-------------------|--------------------|--------------|
| aPourPere | required single | Instance of Homme | |
| aUnLienDeFraternite | multiple | Instance of Humain | |
| dateDeNaissance | required single | String | |
| estMarieA | single | Instance of Humain | |
| habiteA | required single | Instance of Ville | |
| nom | required single | String | |
| prenom | required multiple | String | |

Quelques outils

Protégé: IDE - <https://protege.stanford.edu/>



Quelques outils

WebProtégé: IDE - <http://webprotege.stanford.edu/>

The screenshot displays the WebProtégé IDE interface. The top menu bar includes 'WebProtégé', 'Aero', 'BIBO Ontology', and 'Bioinformatics Web Services ontology'. Below this, a tab bar shows 'Classes', 'Properties', 'Individuals', 'Notes and Discussions', 'Changes By Entity', and 'Project Dashboard'. The main workspace is divided into several panels:

- Classes Panel (Left):** A tree view showing the ontology hierarchy. The 'allergen' class is selected under the 'material entity' class.
- Class description for allergen (Center):** A panel showing the class's metadata and annotations.
 - Display name:** allergen
 - IRI:** http://purl.obolibrary.org/obo/OBI_1110201
 - Annotations:** A table listing various annotations for the class.

| Property | Value | Action |
|-----------------------|---|--------|
| rdfs:label | allergen | [X] |
| definition | A material entity bearing the disposition to cause an allergic reaction | [X] |
| definition source | IEDB | [X] |
| editor preferred term | allergen | [X] |
| example of usage | Birch pollen is an allergen | [X] |
| has curation status | http://purl.obolibrary.org/obo/IAO_0000120 | [X] |
| term editor | IEDB | [X] |
| Enter property name | Enter value | |
 - Properties:** A section for defining instance properties with input fields for 'Enter property name' and 'Enter value'.
- Conditions for allergen (Bottom Center):** A panel showing logical conditions for the class.
 - Equivalent To:** ('material entity' and 'is bearer of some 'disposition to cause an allergic reaction')
 - SubClass Of:** 'material entity'
 - SubClass Of Ancestor Class:** 'continuant', 'owl:Thing', 'independent continuant', 'entity'
- Discussions for allergen (Right):** A panel for discussing the class, with a 'Post new topic...' button.
- Project feed (Bottom Right):** A panel showing project activity, including a message from 'ttania' stating 'started viewing this project Less than one minute ago'.

Quelques Outils

Apache Jena – <https://jena.apache.org/>



A free and open source Java framework for building [Semantic Web](#) and [Linked Data](#) applications.

➤ Get started now!

⬇ Download

RDF

RDF API

Interact with the core API to create and read [Resource Description Framework](#) (RDF) graphs. Serialise your triples using popular formats such as [RDF/XML](#) or [Turtle](#).

ARQ (SPARQL)

Query your RDF data using ARQ, a [SPARQL 1.1](#) compliant engine. ARQ supports remote federated queries and free text search.

Triple store

TDB

Persist your data using TDB, a native high performance triple store. TDB supports the full range of Jena APIs.

Fuseki

Expose your triples as a SPARQL end-point accessible over HTTP. Fuseki provides REST-style interaction with your RDF data.

OWL

Ontology API

Work with models, RDFS and the [Web Ontology Language](#) (OWL) to add extra semantics to your RDF data.

Inference API

Reason over your data to expand and check the content of your triple store. Configure your own inference rules or use the built-in OWL and RDFS [reasoners](#).

Quelques Outils

OWL Validator - S'assurer de la validité, de l'intégrité et de la cohérence des concepts qu'un document OWL exprime.

Valdateur RDF du W3C - <http://www.w3.org/RDF/Validator/>

Check and Visualize your RDF documents

[olde servlet](#)

Enter a URI or paste an RDF/XML document into the text field above. A 3-tuple (triple) representation of the corresponding data model as well as an optional graphical visualization of the data model will be displayed.

Check by Direct Input

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.w3.org/">
    <dc:title>World Wide Web Consortium</dc:title>
  </rdf:Description>
</rdf:RDF>
```

Parse RDF Restore the original example Clear the textarea

Display Result Options:

Triples and/or Graph: Triples Only

Graph format: PNG - embedded

Paste an RDF/XML document into the following text field to have it checked. More options are available in the [Extended interface](#).

Check by URI

Parse URI: Clear the URI

Display Result Options:

Triples and/or Graph: Triples Only

Graph format: PNG - embedded

Enter the URI for the RDF/XML document you would like to check. More options are available in the [Extended interface](#).


Quelques Outils

HermiT OWL Reasoner - <http://www.hermit-reasoner.com/>

● INFORMATION SYSTEMS GROUP ● DEPARTMENT OF COMPUTER SCIENCE ● UNIVERSITY OF OXFORD

Information Systems Group

— Knowledge Representation and Reasoning



About KRR

People

Projects

Tools & Datasets

News & Events

Studentships

Visitors

ISG TOOLS

» Web Tools

» BootOX

» CB

» ContentCVS

» ContentMap

» ELK

» EOLO

» HermiT

» KARMA

» KeywDB

» LogMap

» Module Extractor

» MORE

» Ontology library

» PAGOdA

» PrisM

» RDFox

» RODI

» Requiem

» SamFacet

HermiT OWL Reasoner

The New Kid on the OWL Block

OVERVIEW

HermiT is reasoner for ontologies written using the [Web Ontology Language \(OWL\)](#). Given an OWL file, HermiT can determine whether or not the ontology is consistent, identify subsumption relationships between classes, and much more.

HermiT is the first publicly-available OWL reasoner based on a novel “hypertableau” calculus which provides much more efficient reasoning than any previously-known algorithm. Ontologies which previously required minutes or hours to classify can often be classified in seconds by HermiT, and HermiT is the first reasoner able to classify a number of ontologies which had previously proven too complex for any available system to handle.

HermiT uses direct semantics and passes all OWL 2 conformance tests for direct semantics reasoners.

We have now released [HermiT 1.3.8](#) under the [GNU Lesser General Public License \(LGPL\)](#). The release should be compatible with Java 1.5 or higher. HermiT 1.3.8 uses the [OWL API 3.4.3](#), which is backwards compatible with the OWL API 3.3.x, 3.2.x and 3.1.x, but not backwards compatible with the OWL API 3.0.x.

Since version 1.1, HermiT can handle DL Safe rules and the rules can directly be added to the input ontology in functional style or other OWL syntaxes supported by the OWL API (see [A Syntax for Rules in OWL 2](#)). Note that reasoning with DL Safe rules is incomplete if the ontology contains property chains or transitivity axioms and complex properties are used in the rule bodies.

HermiT is open-source and released under [LGPL](#). All components and source code is included in the project folder of the release.

USING HERMIT

HERMIT AS PROTÉGÉ PLUG-IN

[Protégé 4.3 release](#) is now available and it comes with HermiT pre-installed. Alternatively, the file `org.semanticweb.HermiT.jar` from the latest HermiT release can be copied into Protege's plugin folder. Note that Protégé 4.1 beta (and more recent versions) work with HermiT 1.3.x, whereas Protégé 4.1 alpha works only with the OWL API 3.0.0 and HermiT 1.2.x.

HERMIT FROM THE COMMAND LINE

HermiT provides a command-line interface for common reasoning tasks, including classification and query answering.

HERMIT IN JAVA APPLICATIONS

HermiT supports the `OWLReasoner` interface from the [OWL API](#) and has native support for working with objects such as ontologies and class expression from the OWL API.

NEWS

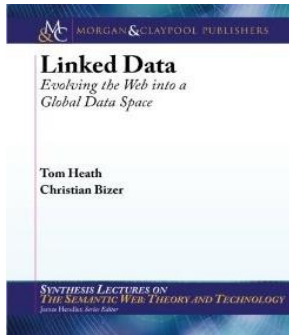
HERMIT 1.3.8 IS NOW AVAILABLE

[Version 1.3.8](#) uses OWL API 3.4.3, and has some bug fixes as described in the readme.

HERMIT NIGHTLY BUILDS AVAILABLE

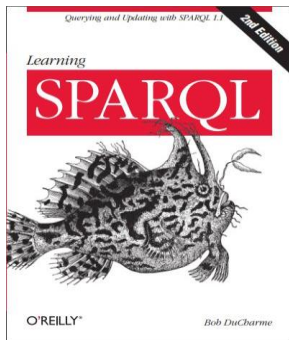
We now provide [nightly builds](#) of HermiT. These

Références



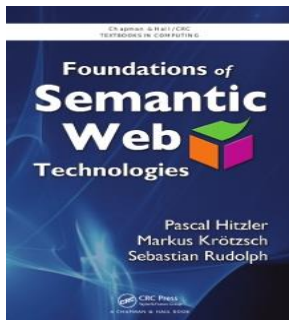
Linked Data: Evolving the Web into a Global Data Space

- ✓ Auteur : Christian Bizer, Tom Heath
- ✓ Éditeur : Morgan & Claypool Publishers
- ✓ Edition : Février 2011 - 136 pages - ISBN 9781608454310



Learning SPARQL : Querying and Updating with SPARQL

- ✓ Auteur : Bob DuCharme
- ✓ Éditeur : O'Reilly Media
- ✓ Edition: Juillet 2013– 386pages -ISBN : 9781449306595



Foundations of Semantic Web Technologies

- ✓ Auteur : Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph
- ✓ Éditeur : CRC Press/Chapman and Hall
- ✓ Edition : 2009 - 455 pages - ISBN : 9781420090505

Références

- W3C – OWL Web Ontology Language
 - ✓ <https://www.w3.org/TR/2004/REC-owl-guide-20040210/>
- INRIA MOOC - Fabien Gandon – Web Sémantique et Web de Données
 - ✓ https://www.canal-u.tv/producteurs/inria/cours_en_ligne/web_semantique_et_web_de_donnees
- Introduction à OWL – Xavier Lacot
 - ✓ http://lacot.org/public/introduction_a_owl.pdf
- Noy et McGuinness - Ontology Development 101: A Guide to Creating Your First Ontology.
 - ✓ https://protege.stanford.edu/publications/ontology_development/ontology101.pdf/
- Haifa Zargayouna – Thèse
 - ✓ <http://www-lipn.univ-paris13.fr/~zargayouna/zargayouna-these.pdf>