

Chapitre VI

DAML+OIL

- 1 Introduction
- 2 Classes
- 3 Propriétés
- 4 Constructions plus complexes

DAML+OIL : Pourquoi définir un nouveau langage ?

- Nécessité de disposer de connaissances ontologiques pour raisonner efficacement sur des connaissances factuelles exprimées en RDF.
- RDF-Schema permet de représenter des connaissances ontologiques... mais est limité :
 - Pas de cardinalité (une personne a exactement un numéro de sécurité sociale, etc.)
 - Pas d'exclusion entre classes (deux classes ne peuvent avoir d'instances communes)
 - Pas de liens plus précis entre classes et entre propriétés (classes équivalentes, transitivité d'une relation, etc)
 - ...

⇒ Nécessité d'un langage plus expressif pour exprimer ces connaissances.

Chapitre VI

DAML+OIL

- 1 Introduction
- 2 Classes
- 3 Propriétés
- 4 Constructions plus complexes

Introduction

Deux initiatives menées en parallèle :

- DARPA Agent Markup Language (**DAML**)
(<http://www.daml.org/>) Ministère de la défense américain.
- Ontology Inference Layer (**OIL**)
(<http://www.ontoknowledge.org/oil/>) Universités et centres de recherche.

Les deux équipes ont fusionné leurs travaux pour proposer

DAML+OIL :

<http://www.daml.org/2001/03/daml+oil-index.html> (2001)

Introduction

Le W3C a publié des notes sur l'utilisation de DAML+OIL pour le web sémantique :

- <http://www.w3.org/TR/daml+oil-reference>
Document de référence.
- <http://www.w3.org/TR/daml+oil-walkthru>
Tutoriel, avec exemples (base de ce cours).

Et a utilisé DAML+OIL comme base à la construction de son propre langage d'ontologies : OWL.

Introduction

- Langage simple à comprendre mais plus complet que RDF-Schema.
Extension de RDF-Schema.
- Syntaxe clairement définie.
- Faciliter des raisonnements automatiques.
- Sémantique formelle claire.
Basée sur une logique de description.

Introduction

Des ontologies en DAML+OIL peuvent être exprimées en RDF (n'importe quel format RDF, habituellement RDF/XML).

Exemple (Espaces de noms usuels)

```
<rdf:RDF
  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/2000/10/XMLSchema#"
  xmlns:daml="http://www.w3.org/2001/10/daml+oil#"
>
```

Dans les exemples, on utilisera, en plus, les espaces de noms :

Exemple (Espaces de noms supplémentaires)

```
xmlns:ex="http://www.exemple.org/exemple#"
xmlns:exd="http://www.exemple.org/exemple-dt#"
xmlns="http://www.exemple.org/exemple#"
```

Introduction

Pour préciser qu'un document RDF est une ontologie DAML+OIL, la class `daml:Ontology` doit être utilisée comme type du document.

Exemple

```
<daml:Ontology rdf:about="">
  <daml:imports
    rdf:resource="http://www.w3.org/2001/10/daml+oil"/>
</daml:Ontology>
```

Introduction

Importation d'ontologie

`imports` permet d'importer des ontologies :

- Les définitions de l'ontologie importée s'appliquent à l'ontologie courante.
- L'importation se fait de façon transitive.
- Attention : L'importation d'une ontologie n'a rien à voir avec les espaces de noms.

Chapitre VI

DAML+OIL

- 1 Introduction
- 2 **Classes**
- 3 Propriétés
- 4 Constructions plus complexes

Définition d'une classe

Utilisation du type `daml:Class`.

Exemple

```
<daml:Class rdf:ID="Animal">
  <rdfs:label>Animal</rdfs:label>
  <rdfs:comment>La classe des animaux.
</rdfs:comment>
</daml:Class>
```

Remarquer l'emploi d'éléments de RDF-Schema.
`daml:Class` est une sous-classe de `rdfs:Class`.

```
<rdfs:Class rdf:ID="Class">
  <rdfs:label>Class</rdfs:label>
  <rdfs:comment>
    The class of all "object" classes
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/
    rdf-schema#Class"/>
</rdfs:Class>
```

Relation de spécialisation entre classes

Exactement de la même façon qu'en RDF-Schema.

Exemple

```
<daml:Class rdf:ID="Male">
  <rdfs:subClassOf rdf:resource="#Animal"/>
</daml:Class>
<daml:Class rdf:ID="Homme">
  <rdfs:subClassOf rdf:resource="#Personne"/>
  <rdfs:subClassOf rdf:resource="#Male"/>
</daml:Class>
<daml:Class rdf:ID="Femme">
  <rdfs:subClassOf rdf:resource="#Personne"/>
  <rdfs:subClassOf rdf:resource="#Femelle"/>
</daml:Class>
```

disjointWith

La propriété `disjointWith` permet d'exprimer que deux classes ne peuvent avoir d'instances communes.

Exemple

```
<daml:Class rdf:ID="Femelle">  
  <rdfs:subClassOf rdf:resource="#Animal"/>  
  <daml:disjointWith rdf:resource="#Male"/>  
</daml:Class>
```

sameClassAs

La propriété `sameClassAs` permet de donner un identificateur "synonyme" à une classe.

Le nom initial ou le synonyme pourront être utilisés indifféremment.

Facilite l'exploitation de documents RDF rédigés sur des ontologies différentes.

Exemple

```
<daml:Class rdf:ID="EtreHumain">  
  <daml:sameClassAs rdf:resource="#Personne"/>  
</daml:Class>
```

Chapitre VI

DAML+OIL

- 1 Introduction
- 2 Classes
- 3 Propriétés
- 4 Constructions plus complexes

Propriétés ressources / données

Les valeurs des propriétés peuvent être

- des ressources (objets)
- des littéraux typés (datatypes)

Lors de la déclaration d'une propriété, on utilisera

- `daml:ObjectProperty` ou
- `daml:DatatypeProperty`

Exemple

```
<daml:ObjectProperty rdf:ID="aPourParent">  
  <rdfs:domain rdf:resource="#Animal"/>  
  <rdfs:range rdf:resource="#Animal"/>  
</daml:ObjectProperty>
```

Plusieurs domain, plusieurs range → intersection des domaines et co-domaines.

Exemple

```
<daml:DatatypeProperty rdf:ID="age">
  <rdfs:domain rdf:resource="#Animal"/>
  <rdfs:range
    rdf:resource="http://www.w3.org/2000/10/XMLSchema
      #nonNegativeInteger"/>
  <rdf:type
    rdf:resource="http://www.w3.org/2001/10/daml+oil
      #UniqueProperty"/>
</daml:DatatypeProperty>
```

Quand une propriété est une UniqueProperty (en plus d'être une ObjectProperty ou DatatypeProperty), au maximum **une** valeur de cette propriété peut être associée à un sujet.

Même syntaxe et même contraintes qu'en RDF-Schema.

Exemple

```
<daml:ObjectProperty rdf:ID="aPourPere">
  <rdfs:subPropertyOf rdf:resource="#aPourParent"/>
  <rdfs:range rdf:resource="#Male"/>
</daml:ObjectProperty>
```

Restriction de propriétés

Principe

La propriété `aPourParent` a pour domaine `Animal` et co-domaine `Animal`.

Comment exprimer qu'un parent d'une `Personne` est une `Personne` ?

(en RDF-Schema, c'est impossible)

Avec DAML+OIL, on peut définir une restriction O sur le co-domaine d'une propriété P quand cette propriété est utilisée avec un sujet faisant partie d'un domaine particulier C .

Cette restriction s'exprime en représentant que C est une sous-classe d'une classe « anonyme » pour laquelle le co-domaine associé à P est O (pour les instances de cette classe anonyme)

Restriction de propriétés

Exemple

Exemple

```
<daml:Class rdf:ID="Personne">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#aPourParent"/>
      <daml:toClass rdf:resource="#Personne"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  ...
```

Restriction de propriétés

Explications

- La Restriction définit une nouvelle classe (anonyme) qui contient tous les objets qui respectent la restriction en question.
- `Personne` étant une `subClassOf` cette classe anonyme, elle doit respecter la restriction.
- Une Restriction peut poser différents types de contraintes. Ici, la restriction est posée sur les triplets dont la propriété est celle précisée dans `onProperty` et dont le sujet est une instance de la classe anonyme : L'objet doit être une instance de la classe précisée dans `toClass` (`Personne`).

Attention. La définition d'un `range` et la définition d'une restriction `toClass` ne sont pas équivalentes : Définition d'une contrainte globale sur la propriété / Définition d'une contrainte sur une propriété quand elle est utilisée sur une classe particulière.

Propriétés et cardinalités

- `cardinality`,
- `maxCardinality` et
- `minCardinality`

permettent de définir des contraintes de

- cardinalité,
- cardinalité maximum,
- cardinalité minimum.

C'est à dire le nombre de fois qu'une propriété peut être utilisée sur un sujet instance d'une classe particulière.

Propriétés et cardinalités

Exemple

```
<daml:Class rdf:ID="Personne"> ...
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#aPourPere"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#telephone"/>
      <daml:minCardinality>1</daml:minCardinality>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

Remarquer les deux syntaxes, qui sont équivalentes.

Propriétés et cardinalités

Exemple

```
<daml:Class rdf:about="#Animal">
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="2">
      <daml:onProperty rdf:resource="#aPourParent"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

Remarquer l'usage du `about` : Ajout de contraintes sur une classe définie ailleurs dans le fichier.

La contrainte définie sur `Animal` s'applique évidemment sur les sous-classes d'`Animal`.

Propriétés et cardinalités

Une contrainte de cardinalité peut être plus précise : Elle peut être exprimée sur les seuls triplets dont l'objet est un type donné.

Exemple

```
<daml:Class rdf:about="#Personne">
  <rdfs:subClassOf>
    <daml:Restriction daml:maxCardinalityQ="1">
      <daml:onProperty rdf:resource="#intervient"/>
      <daml:hasClassQ rdf:resource="#EmploiTempsComplet"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

Propriétés et cardinalités

La propriété `intervient` a une cardinalité maximum de 1 `EmploiTempsComplet` dans la classe `Personne`.

→ Une personne peut être sujet de plusieurs triplets `intervient`, mais au maximum un seul de ces triplets doit avoir pour objet un `EmploiTempsComplet`.

`maxCardinalityQ`, `minCardinalityQ` et `cardinalityQ` doivent toujours être utilisés conjointement avec `hasClassQ`.

Propriétés particulières

Quand une propriété est définie de type `UniqueProperty`, il n'est pas nécessaire de préciser la cardinalité.

Exemple

```
<daml:UniqueProperty rdf:ID="aPourMere">
  <rdfs:subPropertyOf rdf:resource="#aPourParent"/>
  <rdfs:range rdf:resource="#Femelle"/>
</daml:UniqueProperty>
```

Noter que `aPourMere` a une cardinalité de 1 sur tout son domaine. Alors que `aPourPere` a une cardinalité de 1 uniquement sur une partie de son domaine : `Person`.

Propriétés particulières

Définition d'un synonyme d'une propriété : `samePropertyAs`

Exemple

```
<daml:ObjectProperty rdf:ID="aPourMaman">
  <daml:samePropertyAs rdf:resource="#aPourMere"/>
</daml:ObjectProperty>
```

Propriétés particulières

Expression de la propriété inverse par `inverseOf`

Exemple

```
<daml:ObjectProperty rdf:ID="aPourEnfant">
  <daml:inverseOf rdf:resource="#aPourParent"/>
</daml:ObjectProperty>
```

Propriétés particulières

Une propriété est définie de type `UnambiguousProperty` (sous-classe de `ObjectProperty`) pour exprimer la contrainte qu'un objet (d'un triplet) identifie de façon unique un sujet de la propriété.

Quand une propriété est `UniqueProperty`, la propriété inverse est `UnambiguousProperty`.

Exemple

```
<daml:UnambiguousProperty rdf:ID="aPourNumINSEE">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#NumINSEE"/>
</daml:UnambiguousProperty>
```

Propriétés particulières

Expression de la transitivité d'une propriété par le type `TransitiveProperty` (sous-classe de `ObjectProperty`)

Exemple

```
<daml:TransitiveProperty rdf:ID="aPourAncetre"/>
```

Chapitre VI

DAML+OIL

- 1 Introduction
- 2 Classes
- 3 Propriétés
- 4 Constructions plus complexes

Opérations ensemblistes sur les classes

Le `complementOf` d'une classe C est l'ensemble des instances qui ne sont pas des instances de C (ni des sous-classes de C).

Exemple

```
<daml:Class rdf:ID="Voiture">
  <rdfs:subClassOf>
    <daml:Class>
      <daml:complementOf rdf:resource="#Personne"/>
    </daml:Class>
  </rdfs:subClassOf>
</daml:Class>
```

Voiture est une sous-classe du complément de Personne
 ⇒ une instance de Voiture n'est pas une instance de Personne
 ⇒ une ressource ne peut pas être à la fois instance de Voiture et de Personne

Cette notation est équivalente à `disjointWith`.

Opérations ensemblistes sur les classes

Union disjointe

Exemple

```
<daml:Class rdf:about="#Person">
  <daml:disjointUnionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Homme"/>
    <daml:Class rdf:about="#Femme"/>
  </daml:disjointUnionOf>
</daml:Class>
```

Noter l'usage de `rdf:parseType="daml:collection"` (à la place de `rdf:resource="nomClasse"`) qui permet de préciser plusieurs classes.

Opérations ensemblistes sur les classes

Intersection

Exemple

```
<daml:Class rdf:ID="HommeGrand">
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#ChoseGrande"/>
    <daml:Class rdf:about="#Homme"/>
  </daml:intersectionOf>
</daml:Class>
```

HommeGrand est l'intersection de ChoseGrande et Homme. Tout objet qui est instance de ChoseGrande et de Homme est donc aussi (automatiquement) instance de HommeGrand (et inversement).

Opérations ensemblistes sur les classes

Intersection : Utilisation d'une classe anonyme

Exemple

```
<daml:Class rdf:ID="PersonneMariee">
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Personne"/>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#aPourEpoux"/>
    </daml:Restriction>
  </daml:intersectionOf>
</daml:Class>
```

PersonneMariee est (exactement) l'intersection de Personne et de l'ensemble des objets qui sont instances de la classe anonyme regroupant tous les objets qui sont sujet d'une relation aPourEpoux.

Utilisation de datatypes

Définition en XML-Schema

Les datatypes sont définis dans un fichier séparé, en utilisant XML Schema.

Exemple (<http://www.exemple.org/exemple-dt>)

```
<xsd:schema xmlns:xsd=
  "http://www.w3.org/2000/10/XMLSchema#">
  <xsd:simpleType name="ageMajeur">
    <xsd:restriction base="xsd:positiveInteger">
      <xsd:minInclusive value="18"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Utilisation de datatypes

Utilisation dans DAML+OIL

Les datatypes sont utilisés pour poser des restrictions.

Exemple

```
<daml:Class rdf:ID="Adulte">
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Personne"/>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#age"/>
      <daml:hasClass rdf:resource=
        "http://www.exemple.org/exemple-dt#ageMajeur"/>
    </daml:Restriction>
  </daml:intersectionOf>
</daml:Class>
```

Adulte est exactement l'intersection de Personne et de l'ensemble des objets qui sont instances de la classe anonyme regroupant tous les objets qui ont un age qui est un ageMajeur.

Instances dans DAML+OIL

Exemple

```
<daml:ObjectProperty rdf:ID="aPourTaille">
  <rdfs:range rdf:resource="#Taille"/>
</daml:ObjectProperty>
<daml:Class rdf:ID="Taille">
  <daml:oneOf rdf:parseType="daml:collection">
    <Taille rdf:ID="petit"/>
    <Taille rdf:ID="moyen"/>
    <Taille rdf:ID="grand"/>
  </daml:oneOf>
</daml:Class>
```

aPourTaille n'a pas de domain, son co-domaine est Taille.
La classe Taille a exactement 3 instances
<http://www.exemple.org/exemple#petit>, moyen, grand.

Instances dans DAML+OIL

Exemple

```
<daml:Class rdf:ID="GrandeChose">
  <daml:sameClassAs>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#aPourTaille"/>
      <daml:hasValue rdf:resource="#grand"/>
    </daml:Restriction>
  </daml:sameClassAs>
</daml:Class>
```

GrandeChose est exactement l'ensemble des objets qui
aPourTaille grand.

Si utilisation de subclassOf au lieu de sameClassAs, certains
objets pourraient être aPourTaille grand sans être des
GrandeChose (par contre, toute instance de GrandeChose doit
aPourTaille grand).

Thing et Nothing

La classe Thing est super classe de toutes les classes définies. Et la classe Nothing est sous classe de toutes les classes définies : Tout objet est instance de Thing et aucun n'est instance de Nothing.

```
<Class rdf:ID="Thing">
  <rdfs:label>Thing</rdfs:label>
  <rdfs:comment>
    The most general (object) class in DAML.
    This is equal to the union of any class and its complement.
  </rdfs:comment>
  <unionOf rdf:parseType="daml:collection">
    <rdfs:Class rdf:about="#Nothing"/>
    <rdfs:Class>
      <complementOf rdf:resource="#Nothing"/>
    </rdfs:Class>
  </unionOf>
</Class>
<Class rdf:ID="Nothing">
  <rdfs:label>Nothing</rdfs:label>
  <rdfs:comment>the class with no things in it.</rdfs:comment>
  <complementOf rdf:resource="#Thing"/>
</Class>
```