

Chapitre VII

OWL

- 1 Introduction
- 2 Classes
- 3 Propriétés
- 4 Réutilisation d'ontologies
- 5 Classes complexes

OWL

OWL (*Web Ontology Language*) est un langage de description d'ontologies conçu pour la publication et le partage d'ontologies sur le web sémantique.

Le langage est fortement inspiré de DAML+OIL.

- Page du W3C : <http://www.w3.org/2004/OWL/>
- Référence : <http://www.w3.org/TR/owl-ref/>
- Guide : <http://www.w3.org/TR/owl-guide/>
(utilisé comme base à cette présentation)

Chapitre VII

OWL

- 1 Introduction
- 2 Classes
- 3 Propriétés
- 4 Réutilisation d'ontologies
- 5 Classes complexes

OWL : 3 langages

OWL a été conçu pour permettre différents types de raisonnements, plus ou moins complexes.

OWL Lite Langage le plus simple (moins expressif, mais plus facile à programmer, et raisonnements plus rapides).
Hiérarchie de classifications.

Contraintes de cardinalité 0 et 1.

OWL DL Plus grande expressivité, complétude du raisonnement, décidabilité.
Contient tout OWL, avec quelques restrictions (une classe ne peut pas être un individu, etc.).

DL = *Description Logics* (Logiques de description).

Les raisonnements en OWL DL sont ceux qui peuvent être faits dans une logique de description.

Complexité plus élevée que OWL Lite → raisonnements plus lents.

OWL : 3 langages

- OWL Full** Expressivité maximum. Mais pas de garantie de décidabilité.
 Les mécanismes de raisonnement peuvent être très complexes, et la plupart des systèmes ne fournissent qu'une implémentation de OWL Lite ou OWL DL.

OWL Lite \subset OWL DL \subset OWL Full

OWL : Quelques particularités

- Un langage d'ontologies pour le web sémantique → faciliter le raisonnement dans des agents logiciels qui accèdent à des ressources sur le web.
- → Utilisation d'informations distribuées : plusieurs ontologies peuvent être reliées.
- → *monde ouvert* : Une ontologie décrit certaines ressources (ex : B est une sous-classe de A dans une ontologie O), mais ces ressources peuvent être étendues (ex : dans une ontologie O' qui étend O , B est aussi une sous-classe de C).
Monotonie : On peut rajouter de nouvelles informations, mais pas en enlever, ce qui est rajouté peut permettre de nouvelles inférences, mais pas annuler les inférences de l'ontologie initiale.

OWL : Espaces de noms

Exemple (Espaces de noms standard)

```
<rdf:RDF
  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#"
>
```

OWL : Espaces de noms

Exemple (exemples de ce chapitre)

```
<!DOCTYPE rdf:RDF [
  <!ENTITY vin
    "http://www.exemple.com/vin#">
  <!ENTITY nour
    "http://www.exemple.com/nourriture#">
]>

<rdf:RDF
  ...
  xmlns="&vin;"
  xmlns:vin="&vin;"
  xmlns:nour="&nour;"
>
```

Exemple

```
<owl:Ontology rdf:about="">
  <rdfs:comment>Un exemple d'ontologie OWL</rdfs:comment>
  <owl:priorVersion rdf:resource=
    "http://www.exemple.com/old/vin"/>
  <owl:imports rdf:resource=
    "http://www.exemple.com/nourriture"/>
  <rdfs:label>Ontologie sur le vin</rdfs:label>
```

- `priorVersion` lien vers la précédente version de l'ontologie
→ Gestion des évolutions entre versions.
- `imports` permet d'importer une ontologie à compléter.

Chapitre VII

OWL

- 1 Introduction
- 2 **Classes**
- 3 Propriétés
- 4 Réutilisation d'ontologies
- 5 Classes complexes

Déclaration de classe

Toute classe définie dans une ontologie OWL est (implicitement) une sous-classe de `owl:Thing`.

Exemple (dans nous)

```
<owl:Class rdf:ID="Cave"/>  
<owl:Class rdf:ID="Region"/>  
<owl:Class rdf:ID="ChoseConsommable"/>
```

Des informations supplémentaires sur ces classes peuvent être données ailleurs dans le même document (`#Region`) ou dans un autre document (URI complète).

Définition par spécialisation

Exemple (dans nous)

```
<owl:Class rdf:ID="LiquidePotable">  
  <rdfs:subClassOf rdf:resource="#ChoseConsommable" />  
  ...  
</owl:Class>
```

Définition

Une définition de classe contient une liste de restrictions. Ces restrictions posent des conditions sur les objets qui peuvent être instance de la classe.

subClassOf définit une restriction : les seuls objets qui peuvent être instances de la sous-classe sont ceux qui sont instances de la super-classe.

Exemple (dans vin)

```
<owl:Class rdf:ID="Vin">
  <rdfs:subClassOf rdf:resource="&nour;LiquidePotable"/>
  <rdfs:label xml:lang="en">wine</rdfs:label>
  <rdfs:label xml:lang="fr">vin</rdfs:label>
  ...
</owl:Class>
```

Individus

Exemple (dans vin)

```
<Region rdf:ID="Languedoc" />
```

ou (équivalent)

```
<owl:Thing rdf:ID="Languedoc" />
<owl:Thing rdf:about="#Languedoc">
  <rdf:type rdf:resource="#Region"/>
</owl:Thing>
```

Dans la deuxième écriture, un individu est déclaré. Et ailleurs, il est précisé que cet individu est instance de Region.

→ OWL est prévu pour le web : des ontologies peuvent être complétées dans d'autres ontologies.

Classe ou individu ?

Exemple (dans nour)

```
<owl:Class rdf:ID="Raisin">
  ...
</owl:Class>
```

Exemple (dans vin)

```
<owl:Class rdf:ID="RaisinAVin">
  <rdfs:subClassOf rdf:resource="&nour;Raisin" />
</owl:Class>
<RaisinAVin rdf:ID="RaisinCabernetSauvignon" />
```

Comment choisir entre classe et individu ?

Classe ou individu ?

Une classe est un ensemble de choses qui partagent des caractéristiques communes.

Un individu est un membre de cet ensemble.

Un choix du niveau de représentation doit être fait :

- Si *Raisin* représente l'ensemble de tous les (types de) raisins (= cépages), alors *un* de ces cépages doit être une instance de *Raisin*.
- Si *Raisin* représente l'ensemble de tous les (grains de) raisins, alors *Cabernet Sauvignon* doit être une classe (qui contient comme instances tous les grains de *Cabernet Sauvignon*).

Classe ou individu ?

Le choix « sous-classe de » ou « instance de » doit donc se faire en fonction de l'utilisation prévue de l'ontologie.

Par exemple, est-ce que *Vin* représente un ensemble de (types de) vins ou un ensemble de (bouteilles de) vins ?

Dans le premier cas, un crû sera une instance, dans le deuxième une classe.

Le premier cas est adapté pour une ontologie générale sur « le vin ». Le deuxième cas est adapté si on doit gérer un stock de bouteilles.

Chapitre VII

OWL

- 1 Introduction
- 2 Classes
- 3 Propriétés**
- 4 Réutilisation d'ontologies
- 5 Classes complexes

Propriétés

owl:ObjectProperty, owl:DatatypeProperty : exactement comme dans DAML+OIL.

rdfs:subPropertyOf, rdfs:domain, rdfs:range : exactement comme dans RDF-Schema.

Une définition de propriété définit une liste de restrictions, par exemple sur domaine et co-domaine.

Exemple (dans vin)

```
<owl:ObjectProperty rdf:ID="fabriquéAPartirDuRaisin">
  <rdfs:domain rdf:resource="#Vin"/>
  <rdfs:range rdf:resource="#RaisinAVin"/>
</owl:ObjectProperty>
```

Propriétés

Exemple (dans vin)

```
<owl:Class rdf:ID="DescripteurDeVin" />
<owl:Class rdf:ID="CouleurDeVin">
  <rdfs:subClassOf rdf:resource="#DescripteurDeVin" />
  ...
</owl:Class>
<owl:ObjectProperty rdf:ID="aPourDescripteur">
  <rdfs:domain rdf:resource="#Vin" />
  <rdfs:range rdf:resource="#DescripteurDeVin" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="aPourCouleur">
  <rdfs:subPropertyOf rdf:resource="#aPourDescripteur"/>
  <rdfs:range rdf:resource="#CouleurDeVin" />
  ...
</owl:ObjectProperty>
```

Restriction de propriété

(même principe que dans DAML+OIL)

Exemple (dans vin)

```
<owl:Class rdf:ID="Vin">
  <rdfs:subClassOf rdf:resource="&nour;LiquidePotable"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#fabriquéAPartirDuRaisin"/>
      <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Vin est une sous-classe de LiquidePotable et d'une sous-classe (anonyme) qui contient les objets qui ont au moins une propriété fabriquéAPartirDuRaisin → Un vin est fait à partir d'au moins un cépage. N'empêche pas le jus de raisin.

Propriété transitive

$$P(x, y)P(y, z) \Rightarrow P(x, z)$$

Exemple

```
<owl:ObjectProperty rdf:ID="dans">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="&owl;Thing" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>

<Region rdf:ID="Bordeaux">
  <dans rdf:resource="#France" />
</Region>
<Region rdf:ID="France">
  <dans rdf:resource="#Europe" />
</Region>
<Region rdf:ID="Europe"/>
```

Propriété symétrique

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

Exemple

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

Propriété fonctionnelle

$$P(x, y)P(x, z) \Rightarrow y = z$$

Exemple

```
<owl:Class rdf:ID="Millesime" />
<owl:ObjectProperty rdf:ID="aPourMillesime">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Cru" />
  <rdfs:range rdf:resource="#Millesime" />
</owl:ObjectProperty>
```

Un vin est d'un millésime unique (un Cru ne peut être lié par aPourMillesime qu'à un seul Millésime).

Propriété inverse

$$P_1(x, y) \Leftrightarrow P_2(y, x)$$

Exemple

```
<owl:ObjectProperty rdf:ID="aPourFabricant">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="produit">
  <owl:inverseOf rdf:resource="#aPourFabricant" />
</owl:ObjectProperty>
```

Propriété fonctionnelle inverse

$$P(y, x)P(z, x) \Rightarrow y = z$$

Exemple (definitions équivalentes aux précédentes)

```
<owl:ObjectProperty rdf:ID="aPourFabricant" />

<owl:ObjectProperty rdf:ID="produit">
  <rdf:type rdf:resource=
    "&owl;InverseFunctionalProperty"/>
  <owl:inverseOf rdf:resource="#aPourFabricant" />
</owl:ObjectProperty>
```

Restriction de propriété

Exemple

```
<owl:Class rdf:ID="Vin">
  <rdfs:subClassOf rdf:resource="&#223;LiquidePotable"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#aPourFabricant" />
      <owl:allValuesFrom rdf:resource="#Cave" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Vin est une sous-classe d'une classe (anonyme) contenant les objets dont les propriétés aPourFabricant sont des Cave. Le co-domaine de aPourFabricant n'est pas Cave afin de pouvoir utiliser aPourFabricant sur autre chose que des Vin.

Restriction de propriété

Si utilisation de someValuesFrom au lieu de allValuesFrom, un Vin doit avoir au moins un producteur qui est une Cave. (alors que allValuesFrom n'impose pas l'existence d'un producteur)

allValuesFrom : Quantification universelle, vérifié si 0.
someValuesFrom : Quantification existentielle.

Cardinalité

L'expression de contraintes de cardinalité 0 et 1 est possible en OWL Lite. Pour exprimer des cardinalités supérieures, il faut utiliser OWL DL.

Utilisation de `maxCardinality` et `minCardinality` comme dans DAML+OIL.

Restriction de propriété sur une valeur

(OWL DL)

Exemple

```
<owl:Class rdf:ID="Bourgogne">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#niveauSucre" />
      <owl:hasValue rdf:resource="#Sec" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Bourgogne est un vin sec : c'est une sous-classe de la classe anonyme des objets qui sont reliés par `niveauSucre` à la valeur `Sec`.

Chapitre VII

OWL

- 1 Introduction
- 2 Classes
- 3 Propriétés
- 4 Réutilisation d'ontologies
- 5 Classes complexes

Réutilisation d'ontologies

Pour construire une application sur le web sémantique, on peut construire une nouvelle ontologie. . .

Mais concevoir une ontologie de grande taille est très difficile. Il est donc préférable de réutiliser des ontologies existantes, les composer, les étendre pour définir une nouvelle ontologie. Fusionner des ontologies est difficile, mais OWL dispose de fonctions facilitant la réutilisation.

Équivalence

Quand plusieurs ontologies sont utilisées dans une application, il est parfois nécessaire d'indiquer qu'une classe (ou une propriété) d'une ontologie est équivalente à une classe (ou une propriété) d'une autre ontologie : `equivalentClass` et `equivalentProperty`.

Exemple (dans `nour`)

```
<owl:Class rdf:ID="Vin">
  <owl:equivalentClass rdf:resource="#vin;Vin"/>
</owl:Class>
```

Équivalence à une restriction

Exemple

```
<owl:Class rdf:ID="ChosesDuSud">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#dans" />
      <owl:someValuesFrom rdf:resource="#RegionDuSud"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

La classe `ChosesDuSud` contient exactement tous les objets qui sont dans une `RegionDuSud`.

`subClassOf` $\text{ChosesDuSud}(x) \Rightarrow \text{dans}(x,y) \text{RegionDuSud}(y)$

`equivalentClass` $\text{ChosesDuSud}(x) \Leftrightarrow \text{dans}(x,y) \text{RegionDuSud}(y)$

Identité entre individus

L'identité entre individus peut être exprimée explicitement :

Exemple

```
<Vin rdf:ID="VinFavoriDeJean">
  <owl:sameAs rdf:resource="#CabernetDAnjou" />
</Vin>
```

Ou inférée :

Exemple

```
<owl:Thing rdf:about="#ChardonnayBancroft">
  <aPourFabricant rdf:resource="#Bancroft" />
  <aPourFabricant rdf:resource="#Beringer" />
</owl:Thing>
```

aPourFabricant est une propriété fonctionnelle, donc Bancroft = Beringer

Différence entre individus

Exemple

```
<NiveauSucre rdf:ID="Sec" />

<NiveauSucre rdf:ID="Doux">
  <owl:differentFrom rdf:resource="#Sec"/>
</NiveauSucre>
```

Sec est différent de Doux. Si la propriété niveauSucre est une FunctionalProperty, un vin ne peut avoir qu'un niveau de sucre...

Si un vin est marqué à la fois Sec et Doux, le système pourrait donc inférer que Sec = Doux. Si ces deux individus sont définis comme distincts, le système détecte une contradiction.

Différence entre individus

Exemple

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <vin:CouleurDeVin rdf:about="#Rouge" />
    <vin:CouleurDeVin rdf:about="#Blanc" />
    <vin:CouleurDeVin rdf:about="#Rosé" />
  </owl:distinctMembers>
</owl:AllDifferent>
```

Chapitre VII

OWL

- 1 Introduction
- 2 Classes
- 3 Propriétés
- 4 Réutilisation d'ontologies
- 5 Classes complexes

Classes complexes

Mécanisme de formation de classes (en OWL DL) par opérations ensemblistes (intersection, union, complément) sur des classes. Il peut s'agir d'opérations sur des classes existantes ou sur des classes anonymes créées dans l'expression de définition.

Intersection

Exemple

```
<owl:Class rdf:ID="VinBlanc">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Vin" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#aPourCouleur" />
      <owl:hasValue rdf:resource="#Blanc" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Fournit une définition de VinBlanc.

Union

Exemple

```
<owl:Class rdf:ID="Fruit">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#FruitSucré" />
    <owl:Class rdf:about="#FruitNonSucré" />
  </owl:unionOf>
</owl:Class>
```

Complémentaire

Exemple

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

<owl:Class rdf:ID="ChoseNonConsommable">
  <owl:complementOf rdf:resource="#ChoseConsommable"/>
</owl:Class>
```

Complémentaire

Exemple

```
<owl:Class rdf:ID="VinNonFrancais">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Vin"/>
    <owl:Class>
      <owl:complementOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#dans" />
          <owl:hasValue rdf:resource="#France"/>
        </owl:Restriction>
      </owl:complementOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

Définition par énumération

Définition de la liste exhaustive des instances. Aucun autre individu ne peut appartenir à la classe.

Exemple

```
<owl:Class rdf:ID="CouleurDeVin">
  <rdfs:subClassOf rdf:resource="#DescripteurDeVin"/>
  <owl:oneOf rdf:parseType="Collection">
    <owl:CouleurDeVin rdf:about="#Blanc"/>
    <owl:CouleurDeVin rdf:about="#Rosé"/>
    <owl:CouleurDeVin rdf:about="#Rouge"/>
  </owl:oneOf>
</owl:Class>
```

Classes disjointes

Exemple

```
<owl:Class rdf:ID="Pates">
  <rdfs:subClassOf rdf:resource="#ChoseConsommable"/>
  <owl:disjointWith rdf:resource="#Viande"/>
  <owl:disjointWith rdf:resource="#Poisson"/>
  <owl:disjointWith rdf:resource="#Dessert"/>
  <owl:disjointWith rdf:resource="#Fruit"/>
</owl:Class>
```

Les pâtes sont une chose comestible, et cette classe ne peut avoir aucune instance commune avec viande, poisson, etc.