OWL
W3C Ontology Web Language

olivier.corby@sophia.inria.fr

OWL

Un chouette langage d'ontologie pour le web sémantique

Déjà vu

- RDF
- RDFS
- RDF Rules
- SPARQL

OWL

`http://www.w3.org/2001/sw/WebOnt`

Langage d'ontologie pour le Web

Complète les possibilités d'inférences de RDF/S

Raffine les descriptions

RDF

Triplets

- `Resource`
- `Property`
- `Value`

Resource URI, Literal, Blank Node

XML Schema Datatype

RDF

Nommage par URI

```
i:olivier c:author  _:b1

_:b1      rdf:type  c:Book

i:olivier c:name    'Olivier'^^xsd:string
```

RDF/S

```
c:author rdf:type rdf:Property
c:author rdfs:domain c:Document
c:author rdfs:range c:Person

c:author rdfs:subPropertyOf c:creator
```

RDF/S

```
c:Book rdf:type rdfs:Class
c:Book rdfs:subClassOf c:Document

c:Book rdfs:label ‘livre’@fr
```

RDF/S

- rdf:Property
- rdf:type

- rdfs:Class
- rdfs:subClassOf
- rdfs:subPropertyOf
- rdfs:domain
- rdfs:range

Inférences RDF/S

???

Inférences RDF/S

```
?x rdf:type ?class
?class rdfs:subClassOf ?super
→
?x rdf:type ?super
```

Inférences

```
?p1 rdfs:subPropertyOf ?p2
?x ?p1 ?y
→
?x ?p2 ?y
```

Inférences : domain

```
?p rdfs:domain ?class
?x ?p ?y
→
?x rdf:type ?class
```

Inférences : domain (1)

```
c:author rdfs:domain c:Person
i:James c:author i:doc
→
i:James rdf:type c:Person
```

Inférences : domain (2)

```
c:manage rdfs:domain c:Chairman
i:James c:manage i:org
→
i:James rdf:type c:Person
i:James rdf:type c:Chairman
```

Inférences : domain (3)

```
c:speed rdfs:domain c:Mobile
i:James c:speed ‘100’^^xsd:integer
→
i:James rdf:type c:Person
i:James rdf:type c:Chairman
i:James rdf:type c:Mobile
```

Inférences : domain (4)

```
c:engine rdfs:domain c:Car
i:James c:engine i:electric
→
i:James rdf:type c:Person
i:James rdf:type c:Chairman
i:James rdf:type c:Mobile
i:James rdf:type c:Car ???

not(c:Car & c:Person) ???
```

Inférences

```
?p rdfs:range ?class
?x ?p ?y
→
?y rdf:type ?class
```

19

Wanted : inferences ?

20

OWL

Propriétés algébriques des relations
ex: `ex:estMariéAvec` est symétrique.

Correspondances entre deux ontologies
ex: `ex:Voiture` est équivalent à `ex:Car`

Contraintes de cohérence
ex: `ex:Homme` est disjointe de `ex:Femme`

21

OWL

Définition formelles des classes
`ex:Manager(?x)` équivalent à
`?x-(manage)-?y`

Restriction des propriétés et raffinement
`ex:Human`
range `ex:child` is `ex:Human`

22

OWL

OWL issu de

- DARPA DAML (US)
- OIL (EU)

23

OWL

Logique de description

Basée sur RDF

Avec une syntaxe RDF/XML

1. OWL Lite
2. OWL DL
3. OWL Full

24

Semantic Web …

1. XML
2. XML Schema
3. XHTML
4. XPATH
5. XSLT
6. RDF
7. RDF Schema
8. OWL Lite
9. OWL DL
10. OWL Full
11. SPARQL
12. RIF (Rules)
13. GRDDL
14. RDF/A
15. SKOS
16. OWL S (Semantic Web Service)

OWL Lite

Basée sur RDF … moins ε

Classes, propriétés et individus sont *disjoints*

Un individu *ne peut* être aussi une classe

RDF/S vs OWL Lite

```
c:Concept rdfs:subClassOf rdfs:Class
c:Person rdf:type c:Concept
C:James rdf:type c:Person
```

RDF/S : OK
OWL Lite : un individu (`c:Person`) *ne peut* être aussi une classe

RDF/S vs OWL Lite

```
rdf:type rdfs:subPropertyOf rdfs:subClassOf
```

RDF : OK
OWL Lite : impossible

Logique de description

Modèle objet pour la classification de concepts

concept : ensemble d'individus
rôle : relation binaire entre individus

concept et rôle : niveau terminologique (Tbox)
individus : assertion (Abox)

Logique de description

```
X rdf:type aClass => P(X) & Q(X)

X rdf:type aClass <= P(X) & Q(X)

X rdf:type aClass ⇔ P(X) & Q(X)
```

LD

Relation de subsomption : organiser les concepts par niveau de généralité :

un concept A subsume B si

- l'ensemble des individus de B est inclus dans l'ensemble des individus de A :
- A est plus général que B

classification : déterminer la position d'un concept dans une hiérarchie de subsomption

Opérations

Subsomption :

vérifier qu'un concept en subsume un autre. (utile pour valider une classification)

Classification :

placer un concept ou un rôle dans la hiérarchie. (assistance à la construction et l'évolution des ontologies)

Opérations

Satisfiabilité:

vérifier qu'un concept admet des instances (utile pour vérifier la cohérence)

Identification :

retrouver les concepts les plus spécifiques dont un individu est susceptible d'être une instance.

Sémantique

Nc : noms de concepts

Nr : noms de rôles

Définie par une interprétation *I* = (D, I)

D : ensemble d'individus

I : fonction d'interprétation, associe :

Un nom de concept à un ensemble d'individus
$C \in Nc \rightarrow Ci \subseteq D$

Un nom de rôle à une relation binaire
$r \in Nr \rightarrow ri \subseteq D \times D$

Sémantique

Subsomption :

D subsume C ssi :

$Ci \subseteq Di$ pour toute interprétation *I*

OWL

Concepts primitifs vs définis

Les concepts sont définis par des expressions mettant en jeu des concepts et des rôles

Exemples

Une femme est une personne de sexe féminin

$\texttt{Woman} : \texttt{Person} \cap \texttt{Female}$

Une mère est un femme qui a des enfants :

$\texttt{Mother} : \texttt{Woman} \cap \exists \texttt{hasChild.Person}$

OWL Lite (1)

RDF Schema
rdfs:Class
rdfs:subClassOf
rdf:Property
rdfs:subPropertyOf
rdfs:domain
rdfs:range
rdf:type

(In)Equality:
equivalentClass
equivalentProperty
sameAs
differentFrom
AllDifferent
distinctMembers

Property
ObjectProperty
DatatypeProperty
inverseOf
TransitiveProperty
SymmetricProperty
FunctionalProperty
InverseFunctionalProperty

OWL Lite (2)

Property Restriction
Restriction
onProperty
allValuesFrom
someValuesFrom

Class Intersection:
intersectionOf

Datatypes
xsd datatypes

Cardinality
minCardinality (only 0 or 1)
maxCardinality (only 0 or 1)
cardinality (only 0 or 1)

Versioning:
versionInfo
priorVersion
backwardCompatibleWith
incompatibleWith
DeprecatedClass
DeprecatedProperty

Header
Ontology
imports

Annotation Properties:
rdfs:label
rdfs:comment
rdfs:seeAlso
rdfs:isDefinedBy
AnnotationProperty
OntologyProperty

OWL DL et Full

Class Axioms:
oneOf,
dataRange
disjointWith
equivalentClass
(applied to class expressions)
rdfs:subClassOf
(applied to class expressions)

Arbitrary Cardinality:
minCardinality
maxCardinality
cardinality

Boolean Combinations of Class Expressions:
unionOf
complementOf
intersectionOf

Filler Information:
hasValue

OWL Lite/DL/Full

- *OWL Lite* : classification hierarchy and simple constraint features.
- *OWL DL* : maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems. OWL DL includes all OWL language constructs with restrictions such as type separation (a class can not also be an individual or property, a property can not also be an individual or class)..
- *OWL Full* : maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support every feature of OWL Full.

Opérateurs

$C \cap D$
$\exists r.C$
$\forall r.C$
$\geq n\ r.C$
$\leq n\ r.C$

Sémantique

$C \cap D \quad Ci \cap Di$

$\exists r.C \quad \{x \in Di\ /\ \exists y : (x,y) \in ri \ \wedge\ y \in Ci\}$
$\forall r.C \quad \{x \in Di\ /\ \forall y : (x,y) \in ri \Rightarrow y \in Ci\}$

$\geq n\ r.C$
$\{x \in Di/\ |\{y \in Di/(x,y) \in ri \wedge y \in Ci\}| \geq n\}$

$\leq n\ r.C$
$\{x \in Di/\ |\{y \in Di/(x,y) \in ri \wedge y \in Ci\}| \leq n\}$

OWL Racine

La super classe de tout : `owl:Thing`

La classe vide (sans instances) : `owl:Nothing`

owl:Class

```
ex:Human rdf:type owl:Class

owl:Class rdfs:subClassOf rdfs:Class
```

Classe définie

Un humain a des parents humains :

Human(x) ⇒ (parent(x, y) ⇒ Human(y))

```
Human : all parent Human

Human : ∀parent.Human
```

Classe définie : allValuesFrom

```
Human : ∀parent.Human

owl:Class Human
  rdfs:subClassOf
    owl:Restriction
        owl:onProperty parent
        owl:allValuesFrom Human

owl:Restriction : définit une classe
anonyme
```

allValuesFrom : OWL/RDF/XML

```
Human : ∀parent.Human

<owl:Class rdf:about='#Human'>
  <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource='#parent'/>
       <owl:allValuesFrom rdf:resource='#Human'/>
     </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

allValuesFrom : triple

```
Human : ∀parent.Human

c:Human rdf:type owl:Class
c:Human rdfs:subClassOf _:b1
    _:b1 rdf:type owl:Restriction
    _:b1 owl:onProperty c:parent
    _:b1 owl:allValuesFrom c:Human
```

allValuesFrom

```
Human : ∀parent.Human

Marion rdf:type Human
Marion parent Olivier
=>
Olivier rdf:type Human

Gepetto rdf:type Human
Pinocchio parent Gepetto
```
On ne peut rien en déduire

Equivalence

```
Human ⇔ ∀parent.Human

owl:Class Human
    owl:equivalentClass
        owl:Restriction
            owl:onProperty parent
            owl:allValuesFrom Human
```

Equivalence

```
John rdf:type Human
John parent James
=>
James rdf:type Human


John parent James
James rdf:type Human
=>
John rdf:type Human
```

allValuesFrom

Raffiner la signature :

```
Human :   ∀parent.Human

Gorilla : ∀parent.Gorilla

PanTroglodyte :
    ∀parent.PanTroglodyte
```

allValuesFrom : signature

```
owl:Class Human
  rdfs:subClassOf
    owl:Restriction
        owl:onProperty parent
        owl:allValuesFrom Human

owl:Class Gorilla
  rdfs:subClassOf
    owl:Restriction
        owl:onProperty parent
        owl:allValuesFrom Gorilla
```

someValuesFrom

```
Human : ∃parent.Woman

owl:Class Human
  rdfs:subClassOf
    owl:Restriction
        owl:onProperty parent
        owl:someValuesFrom Woman
```

someValuesFrom

```
Human : ∃parent.Woman
```

```
i:Jim rdf:type Human
→
i:Jim c:parent _:b1
_:b1 rdf:type c:Woman
```

Cardinalité

Cardinalité : nombre de valeurs *sémantiquement distinctes* d'une propriété pour un individu

Les humains ont deux parents : (DL)

```
owl:Class Human
    rdfs:subClassOf
        owl:Restriction
            owl:onProperty parent
            owl:cardinality 2
```

Composition

```
owl:Class Human
     rdfs:subClassOf
          owl:Restriction
               owl:onProperty parent
               owl:cardinality 2
     rdfs:subClassOf
          owl:Restriction
               owl:onProperty parent
               owl:someValuesFrom Woman
     rdfs:subClassOf
          owl:Restriction
               owl:onProperty parent
               owl:someValuesFrom Man
```

Cardinalité

```
owl:cardinality 2

owl:maxCardinality 2

owl:minCardinality 2
```

OWL Lite : 0 ou 1

Sous-classe

```
owl:Class Woman
     rdfs:subClassOf Human
     rdfs:subClassOf Female
```

Woman ⇒ Human ∩ Female

Intersection

```
owl:Class Woman
  owl:intersectionOf
     owl:Class Human
     owl:Class Female
```

Définit une équivalence entre Woman et l'intersection de Human et Female

Woman ⇔ Human ∩ Female

Intersection : OWL/RDF/XML

```
<owl:Class rdf:ID='Woman'>
  <owl:intersectionOf parseType='Collection'>
     <owl:Class rdf:about='Human'/>
     <owl:Class rdf:about='Female'/>
  </owl:intersectionOf>
</owl:Class>
```

Intersection : OWL/RDF/XML

```
<owl:Class rdf:ID='Woman'>
  <owl:intersectionOf>
      <rdf:List>
            <rdf:first>
                  <owl:Class rdf:about='Human'/>
            </rdf:first>
            <rdf:rest>
                  <rdf:List>
                        <rdf:first>
                           <owl:Class rdf:about='Female'/>
                        </rdf:first>
                        <rdf:rest
                            rdf:resource='&rdf;nil'/>
                  </rdf:List>
            </rdf:rest>
      </rdf:List>
  </owl:intersectionOf>
</owl:Class>
```

Intersection : RDF triple

```
c:Woman rdf:type owl:Class
c:Woman owl:intersectionOf _:b1
     _:b1 rdf:type rdf:List
     _:b1 rdf:first c:Human
     c:Human rdf:type owl:Class
     _:b1 rdf:rest _:b2
          _:b2 rdf:type rdf:List
          _:b2 rdf:first c:Female
          _:b2 rdf:rest rdf:nil
```

Axiomes

`A rdfs:subClassOf B`

L'extension de A est un sous-ensemble de l'extension de B

`X rdf:type A ⇒ X rdf:type B`

`A owl:equivalentClass B`

même extension : CNS

`X rdf:type A ⇔ X rdf:type B`

Propriétés

`owl:ObjectProperty`

Propriété dont la valeur est un individu (une instance)

`owl:DatatypeProperty`

Propriété dont la valeur est une valeur littérale (integer, string, float, boolean, date)

`owl:AnnotationProperty`

`rdfs:comment rdfs:label`

Ne participent pas aux inférences, purement documentaire

Propriétés

`ex:mother rdfs:subPropertyOf ex:parent`

`rdfs:domain`

`rdfs:range`

Héritées de RDF

Equivalence

p1 owl:**equivalentProperty** p2

X p1 Y ⇔ X p2 Y

Intéressant quand on importe une ontologie :

ex:hasPart owl:equivalentProperty
ns:sous-partie

Inverse

p1 owl:**inverseOf** p2

X p1 Y ⇔ Y p2 X

hasParent owl:inverseOf hasChild

John hasParent Jim ⇔
Jim hasChild John

Symétrie

p rdf:type owl:**SymmetricProperty**

X p Y ⇔ Y p X

ex:sibling rdf:type owl:SymmetricProperty

John ex:sibling Jack ⇔
Jack ex:sibling John

Transitivité

owl:TransitiveProperty

X p Y ∧ Y p Z ⇒ X p Z

ex:partOf rdf:type owl:TransitiveProperty

ex:axis ex:partOf ex:engine
ex:engine ex:partOf ex:car
⇒
ex:axis ex:partOf ex:car

Transitivité

Intérêt pour l'interrogation :

?x ex:partOf ex:car

→

ex:axis, ex:engine

Transitivité

P0 owl:TransitiveProperty
P1 rdfs:subPropertyOf P0
P2 rdfs:subPropertyOf P0

X P1 Y
Y P2 Z
→

Transitivité

```
fratrie owl:TransitiveProperty
frere rdfs:subPropertyOf  fratrie
soeur rdfs:subPropertyOf  fratrie

Jean frere Paul
Paul soeur Anne
→
Jean fratrie Paul
Paul fratrie Anne
→
Jean fratrie Anne
```

Transitivité

```
P0 owl:TransitiveProperty
P1 rdfs:subPropertyOf P0
P2 rdfs:subPropertyOf P0

X P1 Y
Y P2 Z
→
X P0 Z
```

Propriété fonctionnelle

```
ex:husband rdf:type owl:FunctionalProperty
```

Une valeur unique pour une ressource donnée

```
X ex:husband Y
X ex:husband Z
⇒
Y = Z
```

Propriété fonctionnelle inverse

```
ex:motherOf owl:InverseFunctionalProperty
```

Une ressource unique pour une valeur donnée

```
X ex:motherOf Z
Y ex:motherOf Z
⇒
X = Y
```

Individus

Individus identiques :

```
BillClinton owl:sameAs WilliamClinton
```

```
USA president BillClinton
USA president WilliamClinton
```

Cardinality 1 ou 2 ??

Individus

Individus différents :

```
BillClinton owl:differentFrom
     GeorgesDoubleYou
```

Individus

```
<owl:AllDifferent>
      <owl:distinctMembers rdf:parseType="Collection">
            <c:Person rdf:about="#J.Kerry" />
            <c:Person rdf:about="#G.W.Bush" />
            <c:Person rdf:about="#B.Clinton" />
      </owl:distinctMembers>
</owl:AllDifferent>
```

OWL DL & Full

OWL DL & Full : Enumeration

```
owl:Class SouthCity
    owl:oneOf
        ex:Nice
        ex:Marseille
        ex:Montpellier
        ex:Toulouse
```

Tient lieu de définition (CNS, équivalence)

DataRange

```
<owl:DatatypeProperty rdf:ID="tennisGameScore">
<rdfs:range>
<owl:DataRange>
<owl:oneOf>
<rdf:List>
<rdf:first rdf:datatype="&xsd;integer">0</rdf:first>
<rdf:rest> <rdf:List>
<rdf:first rdf:datatype="&xsd;integer">15</rdf:first>
<rdf:rest>
<rdf:List> <rdf:first rdf:datatype="&xsd;integer">30</rdf:first>
<rdf:rest>
<rdf:List> <rdf:first rdf:datatype="&xsd;integer">40</rdf:first>
<rdf:rest rdf:resource="&rdf;nil" /> </rdf:List>
</rdf:rest> </rdf:List>
</rdf:rest> </rdf:List> </rdf:rest> </rdf:List>
</owl:oneOf>
</owl:DataRange>
</rdfs:range>
</owl:DatatypeProperty>
```

Restriction de Valeur

```
owl:Class Human
    rdfs:subClassOf
        owl:Restriction
            owl:onProperty numOfLeg
            owl:hasValue 2
```

Union

```
owl:Class Humanoid
    owl:unionOf
        ex:Man
        ex:Gorilla
        ex:Chimpanzee
        ex:Bonobo
```

CNS (équivalence)

Union

```
owl:Class Worker
      owl:unionOf
            ex:Employee
            ex:Manager

John  rdf:type ex:Employee          John friend Jack
Jack  rdf:type ex:Employee          John friend Jane
Jane  rdf:type ex:Worker            Jack friend Jane
James rdf:type ex:Manager           Jane friend James

select ?x where {
      ?x friend ?y ?y rdf:type ex:Employee
      ?y friend ?z ?z rdf:type ex:Manager
}
```

Union

```
owl:Class Worker
      owl:unionOf
            ex:Employee
            ex:Manager

John  rdf:type ex:Employee            John friend Jack
Jack  rdf:type ex:Employee            John friend Jane
Jane  rdf:type ex:Worker(Employee)    Jack friend Jane
James rdf:type ex:Manager             Jane friend James

select ?x where {
      ?x friend ?y ?y rdf:type ex:Employee
      ?y friend ?z ?z rdf:type ex:Manager
}
```

Union

```
owl:Class Worker
      owl:unionOf
            ex:Employee
            ex:Manager

John  rdf:type ex:Employee            John friend Jack
Jack  rdf:type ex:Employee            John friend Jane
Jane  rdf:type ex:Worker(Manager)     Jack friend Jane
James rdf:type ex:Manager             Jane friend James

select ?x where {
      ?x friend ?y ?y rdf:type ex:Employee
      ?y friend ?z ?z rdf:type ex:Manager
}
```

Complement

```
owl:Class Invertebrate
    owl:complementOf
          Vertebrate
```

L'ensembles des individus qui ne sont pas membre de la classe (`Vertebrate`)

```
owl:Class Invertebrate
    rdfs:subClassOf Animal
    owl:complementOf Vertebrate
```

Boolean operations

unionOf : OR

intersectionOf : AND

complementOf : NOT

Disjoint

```
owl:Class Man
    owl:disjointWith
          Woman
```

Un individu ne peut être membre des deux classes à la fois

Les extensions sont disjointes

Autre

```
<owl:imports
    rdf:resource='#onto.owl'/>

owl:deprecatedClass

owl:deprecatedProperty
```

The End

Si vous avez compris tout ce que je viens de dire … c'est que je me suis mal exprimé