OWL

Web Ontology Language

http://www.w3.org/TR/owl2-overview/

Langage de référence pour définir des ontologies

Plus riche que RDFS : définition de classes à l'aide de constructeurs, caractéristiques des propriétés, contraintes de cardinalités...

Extension de RDFS

OWL est basé sur les *logiques de description*

plusieurs syntaxes, et plusieurs "profils"' plus ou moins complexes

OWL

une ontologie OWL est un ensemble d'énoncés : les *axiomes*

une *entité* désigne un objet (du monde réel)

présentation de OWL2 :

https://www.w3.org/TR/2012/REC-owl2-primer-20121211/

https://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211/

OWL

Classes

* définition d'une classe, d'une entité

```
ex:Man rdf:type owl:Class
ex:John rdf:type ex:Man
```

* sous-classes

```
rdfs:subclassOf
```

* equivalence

owl:equivalentClass

```
:Person  rdf:type           owl:Class .
:Person  owl:equivalentClass :Human .
```

* définition de classes disjointes

```
[ ]  rdf:type     owl:AllDisjointClasses ;
     owl:members  ( :Woman  :Man ) .
```

Propriétés

* type
  - **owl:ObjectProperty**
  - **owl:DatatypeProperty**
  - **owl:AnnotationProperty**

* domaine, rang, sous-propriétés : rdfs
  - rdfs:domain
  - rdfs:range
  - rdfs:subPropertyOf

Caractéristiques des propriétés

* symétrique / antisymétrique

owl:SymmetricProperty

ex:est_marié_avec rdf:type owl:SymmetricProperty

owl:AsymmetricProperty

ex:a_pour_enfant rdf:type owl:AsymmetricProperty

* réflexive / antiréflexive

owl:ReflexiveProperty

owl:IrreflexiveProperty

* transitive

owl:TransitiveProperty

ex:ancetre rdf:type owl:TransitiveProperty

OWL

*fonctionnelle

owl:FunctionalProperty

ex:est_mere_d**e rdf:type** owl:FunctionalProperty

p est fonctionnelle si: $p(X,Y) \wedge p(X,Z) \rightarrow Y=Z$

owl:InverseFunctionalProperty

$p(Y,X) \wedge p(Z,X) \rightarrow Y=Z$

* propriétés inverses

owl:inverseOf

ex:a_pour_enfant **owl:inverseOf** ex:a_pour_parent

* propriétés disjointes

owl:propertyDisjointWith

:hasParent **owl:propertyDisjoint**With :hasSpouse .

p1 et p2 sont disjointes si **:** $p1(X,Y) \rightarrow \neg p2(X,Y)$

Comparaison d'entités

* different - sameAs

:John **owl:differentFrom** :Bill .

:James **owl:sameAs** :Jim .

utilisé pour relier des ressources de bases différentes :
ex:Lille owl:sameAs dbpedia_fr:Lille

Définition de Classses complexes:

Constructeurs : union intersection complément

Union

```
:Parent  owl:equivalentClass  [
                    rdf:type     owl:Class ;
                   owl:unionOf  ( :Mother :Father )
                                ] .
```

Intersection

```
:Mother  owl:equivalentClass  [
                    rdf:type          owl:Class ;
                    owl:intersectionOf  ( :Woman :Parent )
                                ] .
```

Complément

```
:ChildlessPerson  owl:equivalentClass  [
                    rdf:type          owl:Class ;
                    owl:intersectionOf  ( :Person
                     [ rdf:type         owl:Class ;
                       owl:complementOf  :Parent ] )
                                                  ] .
```

la classe :Childless est l'ensemble des entités de la classe :Person qui ne sont pas dans la classe :Parent

Restrictions

someValuesFrom
allValuesFrom
hasValue

someValuesFrom

un parent est une personne qui a au moins un enfant :

```
:Parent  owl:equivalentClass  [
                    rdf:type          owl:Restriction ;
                    owl:onProperty     :hasChild ;
                    owl:someValuesFrom  :Person
                              ] .
```

someValuesFrom

un parent est une personne qui a au moins un enfant :

:Parent owl:equivalentClass [
rdf:type **owl:Restriction** ;
owl:onProperty :hasChild ;
owl:someValuesFrom :Person
] .

Une restriction (**owl:Restriction**) applique une contrainte (**owl:someValuesFrom :Person**) à une propriété (**owl:onProperty**)

La classe :Parent est l'ensemble des ?x pour lesquels il existe un ?y tel que :

?y rdf:type :Person
?x :hasChild ?y

parent(X) si et seulement si : $\exists Y$ hasChild(X,Y) $\wedge$ person(Y)

allValuesFrom

définition de la classe :HappyParent : ensemble des personnes dont tous les enfants sont dans la classe :Happy

:HappyParent owl:equivalentClass [
rdf:type **owl:Restriction** ;
owl:onProperty :hasChild ;
owl:allValuesFrom :Happy
] .

:HappyParent est l'ensemble des entités telles que toutes les valeurs associées par la propriété :hasChild sont des instances de la classe :Happy

ou : happyParent(X) si et seulment si $\forall Y$ (hasChild(X,Y) $\rightarrow$ happy(Y))

hasValue

Les parents de :Bob :

```
:Parent  owl:equivalentClass  [
                    rdf:type          owl:Restriction ;
                    owl:onProperty     :hasChild ;
                    owl:hasValue  :Bob
                                                        ] .
```

Cardinalités

restriction pour définir des contraintes de cardinalités
exemple: au moins un enfant / exactement un enfant / au plus deux enfants

:John a trois enfants :

```
:John  rdf:type  [
            rdf:type        owl:Restriction ;
            owl:cardinality  "3"^^xsd:nonNegativeInteger ;
            owl:onProperty   :hasChild
] .
```

la restriction définit l'ensemble des ?x pour lesquels il existe exactement 3 ?y tels que:

?x :hasChild ?y

minCardinality

```
:John  rdf:type  [
            rdf:type        owl:Restriction ;
            owl:minCardinality  "1"^^xsd:nonNegativeInteger ;
            owl:onProperty   :hasChild
] .
```

maxCardinality

```
:John  rdf:type  [
            rdf:type        owl:Restriction ;
            owl:maxCardinality  "5"^^xsd:nonNegativeInteger ;
            owl:onProperty   :hasChild
] .
```

:

qualifiedCardinality

:John a trois enfants qui sont des parents:

```
:John  rdf:type  [
            rdf:type        owl:Restriction ;
            owl:qualifiedCardinality  "3"^^xsd:nonNegativeInteger ;
            owl:onProperty   :hasChild
            owl:onClass              :Parent
] .
```

la restriction définit l'ensemble des ?x pour lesquels il existe exactement 3 ?y tels que:

```
?x :hasChild ?y
?y rdf:type :Parent
```

Profils OWL et Inférence

Plusieurs sémantiques, plusieurs profils

chaque profil correspond à un sous-ensemble des primitives

- plus ou moins expressif
- inférences plus ou moins complexes
- OWL 2
  - OWL DL
    - OWL 2 EL
    - OWL 2 QL
    - OWL 2 RL

Sémantique directe basée sur la logique de descriptions.

Sémantique basée sur la sémantique de RDFS

Exemple (extraits), source : https://www.w3.org/TR/2012/REC-owl2-primer-20121211/

```
@prefix : <http://example.com/owl/families/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:hasSpouse    rdf:type  owl:SymmetricProperty .
:hasRelative  rdf:type  owl:ReflexiveProperty .
:parentOf     rdf:type  owl:IrreflexiveProperty .
:hasAncestor  rdf:type  owl:TransitiveProperty .

:hasHusband   rdf:type  owl:FunctionalProperty .
:hasHusband   rdf:type  owl:InverseFunctionalProperty .

:hasWife rdf:type          owl:ObjectProperty .
:hasWife rdfs:domain       :Man ;
         rdfs:range        :Woman .
:hasWife rdfs:subPropertyOf :hasSpouse .

:hasSon     owl:propertyDisjointWith  :hasDaughter.

:hasFather  rdfs:subPropertyOf       :hasParent.
:hasParent owl:inverseOf           :hasChild .
:hasParent owl:propertyDisjointWith  :hasSpouse .

:hasGrandparent  owl:propertyChainAxiom  ( :hasParent  :hasParent ) .
:hasUncle        owl:propertyChainAxiom  ( :hasFather  :hasBrother ) .

:hasAge  rdf:type            owl:DatatypeProperty .
:hasAge  rdf:type            owl:FunctionalProperty .
:hasAge  rdfs:domain           :Person ;
         rdfs:range          xsd:nonNegativeInteger .

:hasChild  owl:equivalentProperty  otherOnt:child .
:hasChild  rdf:type   owl:AsymmetricProperty .

:Woman   rdfs:subClassOf :Person .
:Mother  rdfs:subClassOf :Woman .
:Mother  owl:equivalentClass  [
         rdf:type owl:Class ;
         owl:intersectionOf  ( :Woman :Parent )
] .

:Person  rdf:type  owl:Class .
:Person  owl:equivalentClass :Human .
:Person  rdfs:comment    "Represents the set of all people.".
```

```
:Parent  owl:equivalentClass [
        rdf:type     owl:Class ;
        owl:unionOf  ( :Mother :Father )
        ] .
:Parent  owl:equivalentClass  [
        rdf:type           owl:Restriction ;
        owl:onProperty      :hasChild ;
        owl:someValuesFrom  :Person
        ] .
:Grandfather  rdfs:subClassOf  [
        rdf:type owl:Class ;
        owl:intersectionOf  ( :Man  :Parent )
        ] .
:HappyPerson
        owl:equivalentClass  [
        rdf:type owl:Class ;
        owl:intersectionOf  (
                [ rdf:type owl:Restriction ;
                  owl:onProperty  :hasChild ;
                  owl:allValuesFrom   :Happy      ]
                [ rdf:type  owl:Restriction ;
                   owl:onProperty      :hasChild ;
                    owl:someValuesFrom  :Happy     ]
                )    ] .
```

```
:JohnsChildren  owl:equivalentClass  [
                rdf:type owl:Restriction ;
                owl:onProperty  :hasParent ;
                owl:hasValue    :John
                ] .
:NarcisticPerson owl:equivalentClass  [
                 rdf:type  owl:Restriction ;
                owl:onProperty  :loves ;
                owl:hasSelf     "true"^^xsd:boolean .
                ] .
:MyBirthdayGuests  owl:equivalentClass  [
 rdf:type   owl:Class ;
 owl:oneOf  ( :Bill  :John  :Mary )
                ] .

:Teenager  rdfs:subClassOf
    [ rdf:type           owl:Restriction ;
      owl:onProperty       :hasAge ;
      owl:someValuesFrom
       [ rdf:type            rdfs:Datatype ;
         owl:onDatatype       xsd:integer ;
      owl:withRestrictions (  [ xsd:minExclusive     "12"^^xsd:integer ]
                              [ xsd:maxInclusive     "19"^^xsd:integer ])
       ]
        ] .
```

```
:Father rdfs:subClassOf  [
  rdf:type           owl:Class ;
  owl:intersectionOf  ( :Man  :Parent )
] .
:ChildlessPerson      owl:equivalentClass  [
  rdf:type           owl:Class ;
  owl:intersectionOf  ( :Person  [ owl:complementOf  :Parent ] )
] .

[ ] rdf:type     owl:AllDisjointClasses ;
   owl:members  ( :Woman  :Man ) .

:personAge  owl:equivalentClass
 [ rdf:type       rdfs:Datatype;
   owl:onDatatype xsd:integer;
   owl:withRestrictions (
      [ xsd:minInclusive "0"^^xsd:integer ]
      [ xsd:maxInclusive "150"^^xsd:integer ]
   )
 ] .
:Mary rdf:type   :Person .
:Mary rdf:type   :Woman .
:Mary owl:sameAs otherOnt:MaryBrown .
:James owl:sameAs :Jim .
```

```
:Jack  rdf:type         [
  rdf:type           owl:Class ;
  owl:intersectionOf  ( :Person
                        [ rdf:type         owl:Class ;
                          owl:complementOf  :Parent     ]
                      )
] .

:John  owl:sameAs        otherOnt:JohnBrown .
:John  rdf:type          owl:NamedIndividual .
:John  rdf:type          :Father .
:John  :hasWife          :Mary .
:John  owl:differentFrom :Bill .
:John  :hasAge           51 .
:John  rdf:type  [
  rdf:type                 owl:Restriction ;
  owl:qualifiedCardinality  "3"^^xsd:nonNegativeInteger ;
  owl:onProperty           :hasChild ;
  owl:onClass              :Parent
] .
:John  rdf:type  [
  rdf:type         owl:Restriction ;
  owl:cardinality  "5"^^xsd:nonNegativeInteger ;
  owl:onProperty   :hasChild
] .
```

```
[] rdf:type owl:NegativePropertyAssertion ;
   owl:sourceIndividual :Bill ;
   owl:assertionProperty :hasWife ;
   owl:targetIndividual :Mary .

[] rdf:type owl:NegativePropertyAssertion ;
   owl:sourceIndividual :Bill ;
   owl:assertionProperty :hasDaughter ;
   owl:targetIndividual :Susan .

[] rdf:type owl:NegativePropertyAssertion ;
   owl:sourceIndividual :Jack ;
   owl:assertionProperty :hasAge ;
   owl:targetValue 53 .
```