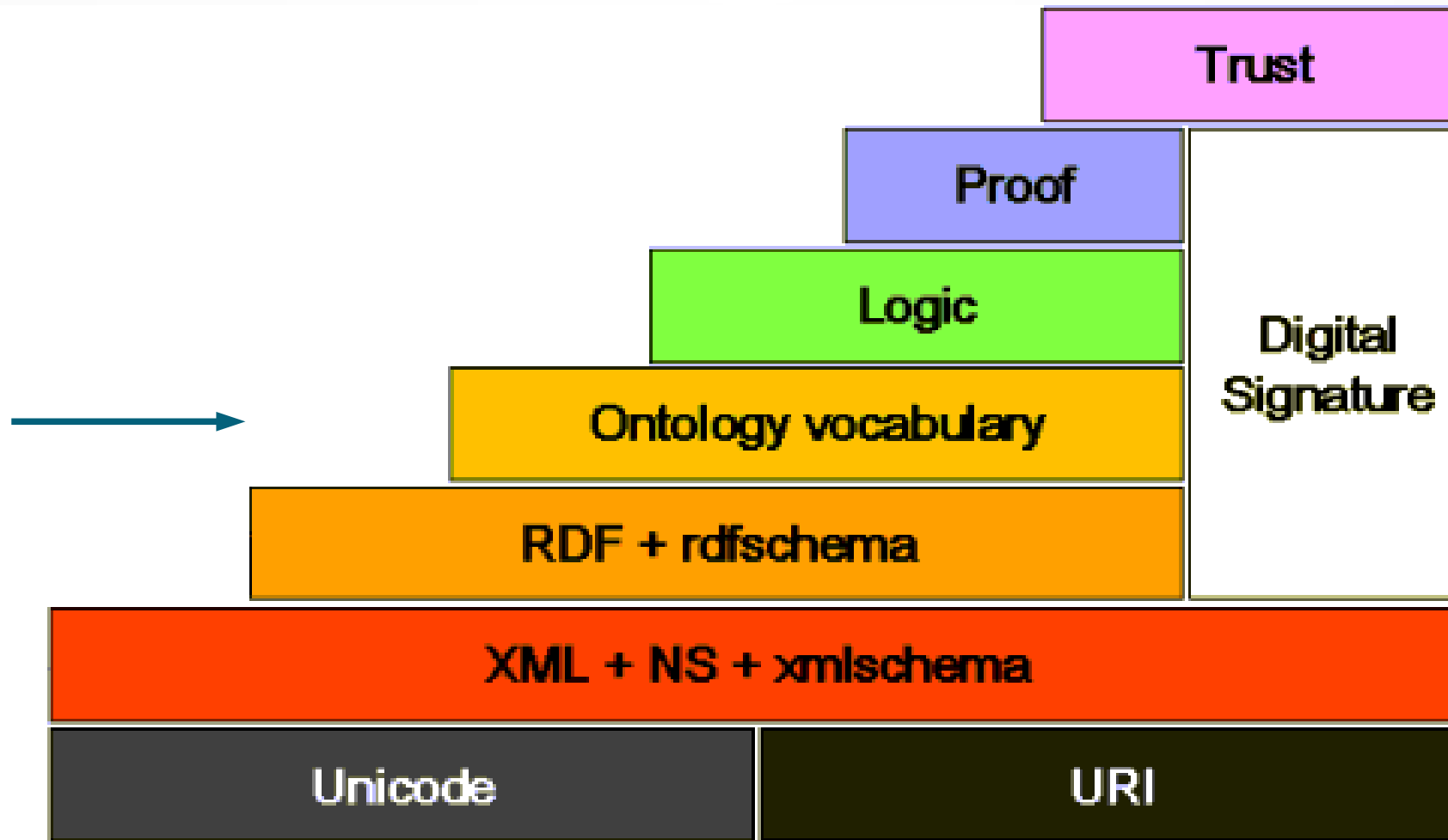


Le web sémantique

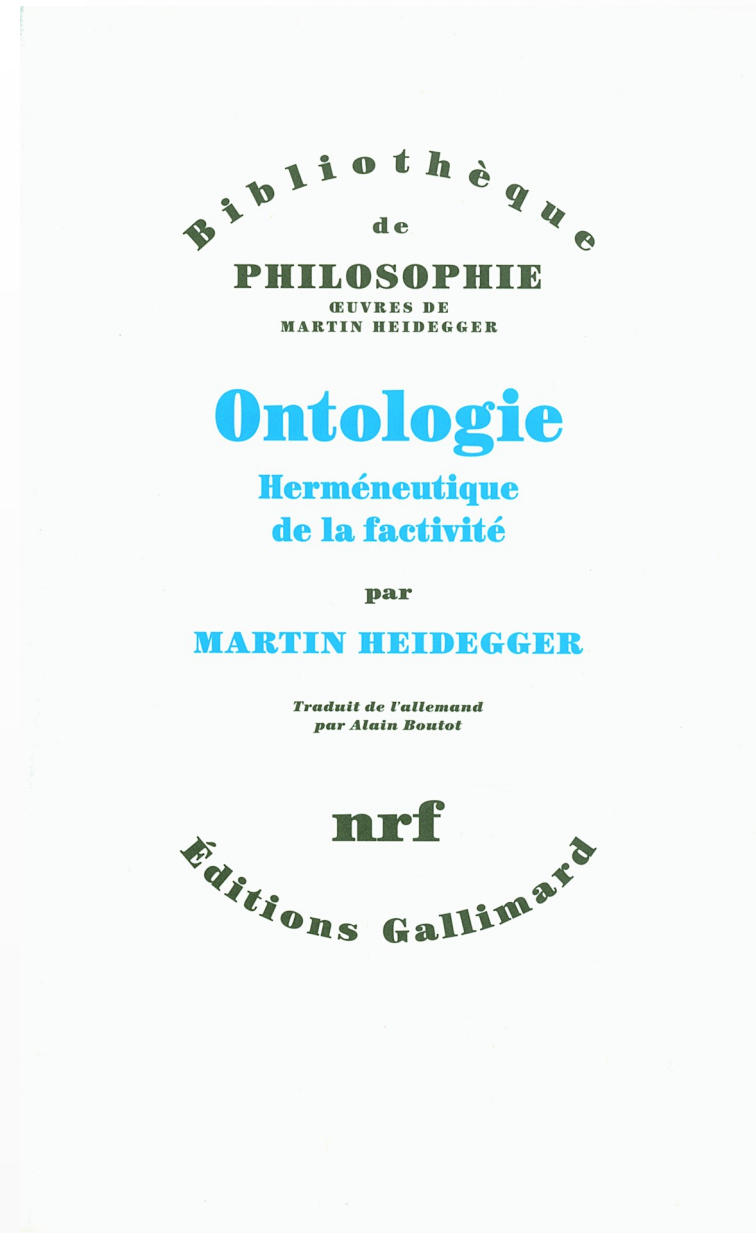
OWL

Semantic Web tower (Berners-Lee)



Ontologie (au singulier)

- Truc de philosophes



Les ontologies

- Truc d'informaticiens
- « spécification explicite d'une conceptualisation » [Gruber]
- « Une conceptualisation est une vue abstraite et simplifiée du monde que l'on veut représenter » [Gruber]

[Gruber], *Towards Principles for the Design of Ontologies Used for Knowledge Sharing in Formal Ontology in Conceptual Analysis and Knowledge Representation*, Kluwer Academic Publishers, 1993

Les ontologies

- Représentation formelle d'un domaine de connaissances
- Schéma entités/associations
- Un graphe RDF = une ontologie
- Des faits + des déductions pour déduire d'autres faits
- Comme un schéma de BD / un diagramme de classes, mais plus général

RDF+RDFS

- RDF seul n'est pas assez riche pour décrire un domaine de connaissance (une ontologie)
- RDF+RDFS permettent de définir des vocabulaires plus vastes
- <https://schema.org> est décrit entièrement via RDF+RDFS
- Les slides du cours pourraient être formellement décrits via RDF+RDFS (axiomes)

RDF+RDFS : limites

- On n'a pas la puissance d'un diagramme de classes / schéma de BD, et d'autres choses pourraient être ajoutées :
 - Comment décrire des cardinalités ? Un roman a entre 0 et n auteurs, mais un et un seul titre, et un romancier a écrit entre 1 et n romans
 - Comment décrire des relations inverses ? Si :a is :child of :b, alors :b is :parent of a
 - Comment décrire des négations ? Un chien n'est jamais un chat, si a est un chien, alors a n'est pas un chat

OWL : Web Ontology Language

- Aller encore plus loin que RDFS en terme d'expressivité
- Recommandation W3C
- OWL : 2003
- OWL 2 : 2009, mise à jour en 2014
- Actuellement encore un domaine de recherche
 - Logiques de description
- Pas une surcouche de RDFS, peut être utilisé indépendamment ou non

OWL : Web Ontology Language

- Pas un langage mais une famille de langages
- Plusieurs langages :
 - Modèles très puissants... mais indécidables
 - Modèles plus simples, décidables, mais à la complexité élevée
 - Modèles très limités, mais algos polynomiaux
 - Assez puissant pour représenter un schéma de BD ou un diagramme de classes

Plusieurs langages pour plusieurs besoins

- Effectuer des recherches / des requêtes dans une base de connaissance
 - Moteurs de recherche
 - Pas besoin d'une expressivité exceptionnelle
 - Besoin de chercher dans des bases de connaissance de taille colossale

Plusieurs langages pour plusieurs besoins

- Effectuer des raisonnements sur des bases de connaissances
 - Déduire / découvrir de nouvelles connaissances
 - Data mining
 - Faire des preuves formelles
 - Preuve par l'absurde : ajout d'une hypothèse puis recherche d'une incohérence
- Besoin de plus d'expressivité
- Mais nécessaire d'être calculable

Preuve par l'absurde

- La chose qui m'a aboyé dessus ne peut pas être un chat, la preuve
- Faits connus :
 - Les chiens ne sont pas des chats (non exprimable en RDFS)
 - `:aboieSur rdfs:domain :Chien .`
 - `_:x :aboieSur :moi .` # Quelque chose m'a aboyé dessus
- Est-ce que `_:x` peut être un chat ? Ajout d'hypothèse
 - `_:x a :Chat .`
- Dédutions :
 - `_:x :aboieSur :moi . :aboieSur rdfs:domain :Chien \rightarrow _:x a :Chien`
 - `_:x a :Chien, :Chat $\rightarrow \perp$ # incohérence`

Plusieurs langages pour plusieurs besoins

- Décrire formellement un domaine de connaissance
 - Pour des raisons contractuelles (cahier des charges)
 - Pour faciliter la communication entre acteurs d'un même domaine
- Pas forcément besoin de pouvoir faire du raisonnement / des requêtes
- Besoin de pouvoir être aussi expressif que possible

OWL 2 et ses profils

- D'un côté OWL 2 complet (OWL 2 / Full)
 - Très expressif
 - Non décidable → pas de raisonnement possible
- De l'autre des « profils »
 - OWL 2 / EL
 - OWL 2 / QL
 - OWL 2 / RL
- On peut aussi définir son propre sous-ensemble

OWL 2 / EL

- Expression Language
- Le plus expressif des 3
- Pour les ontologies qui ont beaucoup de classes / propriétés
- En temps polynomial :
 - Cohérence (la base est-elle forcément fausse ?),
 - Dédutions de classes (est-ce que `C1 subClassOf C2` ?)
 - Dédutions d'instances (est-ce que `x1 rdf:type C1` ?)

OWL 2 / QL

- Query Language
- En temps logarithmique
- Même pouvoir expressif qu'un schéma de BD ou un diagramme UML
 - Relativement peu de classes / propriétés, énormément d'instances
 - Niveau d'expressivité de SQL (plus ou moins)
 - Idéal pour convertir une BD relationnelle en ontologie

OWL 2 / RL

- Reasoning Language
- Systèmes à base de règles
- Conçu pour enrichir sémantiquement les ontologies décrites avec RDFS

OWL 2

- Langage très vaste
- Sous-ensembles
- Indépendant de mais compatible avec RDFS
 - Beaucoup de propriétés et concepts redondants :
 - SubClassOf
 - SubObjectPropertyOf
 - ClassAssertion (équivalent de rdf:type)
 - Etc.
- On se focalisera sur ce qui diffère, pour « enrichir » RDFS

Hypothèse du monde ouvert

- Dans les ontologies, souvent, ce qui n'est pas explicitement vrai peut être soit vrai, soit faux
- Si on ne dit pas explicitement que les chats ne sont pas des chiens, et que x est un chat, alors il est tout à fait possible que x soit aussi un chien

Équivalences de classes

- Synonymie
- Tous les A sont des B, et tous les B sont des A : A et B sont synonymes

`:aaa rdfs:subClassOf :bbb`

`:bbb rdfs:subClassOf :aaa`

`:aaa owl:equivalentClass :bbb`

Équivalence d'instances

- « Astre du jour » et « soleil » sont des « corps célestes »
- Ils sont synonymes
- On ne peut pas écrire ça en RDFS

```
:astreDuJour a :CorpsCéleste .
```

```
:soleil a :CorpsCéleste .
```

```
:astreDuJour owl:sameAs :soleil .
```

```
:astreDuJour = :soleil . # Notation 3
```

```
:astreDuJour :xxx :yyy → :soleil :xxx :yyy
```

Classes disjointes

- On ajoute la négation : les A ne sont pas des B
- Qui dit négation dit incohérence possible

`:Cat rdfs:subClassOf :Animal`

`:Dog rdfs:subClassOf :Animal`

`:Cat owl:disjointWith :Dog`

`:freyja a :Cat, :Dog → :freyja a owl:Nothing`

Individus différents

- Le soleil et la lune sont des corps célestes
- Le soleil n'est pas la lune

`:soleil a :CorpsCéleste .`

`:lune a :CorpsCéleste .`

`:soleil owl:differentFrom :lune .`

Classes complexes

- Notation ensembliste pour décrire des classes anonymes
 - `owl:objectIntersectionOf`
 - `owl:objectUnionOf`
 - `owl:objectComplementOf`

Classes complexes : intersection

- La classe de tout ce qui est à la fois un film et un livre

```
[ owl:objectIntersectionOf (schema:Book schema:Movie) ]
```

- Une « adaptation cinématographique » est à la fois un livre et un film

```
:Adaptation owl:equivalentClass [  
    owl:objectIntersectionOf (schema:Book schema:Movie)  
] .
```

Classes complexes : union

- Un animal domestique, c'est soit un chat, soit un chien, soit un lapin

```
:Domestic rdfs:subClassOf [ owl:objectUnionOf  
(:Cat :Dog :Rabbit) ] .
```

```
:freyja a :Cat → :freyja a :Domestic
```

Classes complexes : négation

- Un extraterrestre, c'est quelqu'un qui n'est pas terrien

```
:Terrien rdfs:subClassOf :Indivudal .
```

```
:Alien rdfs:subClassOf :Individua1 ,
```

```
[ owl:complementOf (:Terrien) ] .
```

Classes complexes : négation

- Deux classes a et b sont disjointes → a est sous-classe de la négation de b

```
:Cat owl:disjointWith :Dog .
```

```
→ :Cat rdfs:subClassOf [ owl:complementOf (:Dog) ] .
```

Tous les chats sont des non-chiens : aucun chat n'est un chien

Classes complexes : négation

- Attention : sous-classe et pas « classes équivalentes »
`:Cat owl:equivalentClass [owl:complementOf (:Dog)] .`
- Tous les chats sont des non-chiens : aucun chat n'est un chien
- Tous les non-chiens sont des chats
- Totor le castor n'est pas un chien
- Donc Totor le castor est un chat (!)

Classes complexes : énumération

- Définition en extension
- Un mois de 30 jours c'est soit avril, soit juin, soit septembre, soit novembre

```
:Mois30 rdfs:subClassOf [  
    owl:oneOf (:avril :juin :septembre :novembre) ] .
```

```
:x a :Mois30 ;  
    owl:differentFrom :avril, :juin, :septembre .
```

```
→ :x owl:sameAs :novembre .
```

Propriétés inverses

- Si a est parent de b, alors b est enfant de a
- Parent et enfant sont des propriétés inverses

```
:parent a rdf:Property .
```

```
:child a rdf:Property ;
```

```
    owl:inverseOf :parent .
```

```
:aaa :parent :bbb .
```

```
→ :bbb :child :aaa .
```

Propriétés inverse

```
:r1 rdfs:domain :C1 ;  
    rdfs:range :C2 .
```

```
:r2 owl:inverseOf :r1 .
```

```
→ :r1 a rdf:Property .
```

```
→ :r2 a rdf:Property ;  
    rdfs:domain :C2 ;  
    rdfs:range :C1 .
```

Propriétés fonctionnelles

- f est une fonction ssi pour une valeur x donnée, il existe au maximum une seule valeur $f(x)$
 - f est une propriété de cardinalité maximale 1
 - Un livre a un seul auteur, un auteur peut avoir plusieurs livres
 - `aPourAuteur` est fonctionnelle
 - `estAuteurDe` n'est pas fonctionnelle
- `:aPourAuteur a owl:FunctionalProperty .`

Propriétés fonctionnelles

```
:aPourAuteur a owl:FunctionalProperty .
```

```
:swag :aPourAuteur :elmore,  
      :x .
```

```
:x owl:differentFrom :elmore .
```

→ \perp # incohérence

Propriétés fonctionnelles

- Si une propriété est fonctionnelle, son inverse est « fonctionnelle inverse »
 - aPourAuteur n'est pas fonctionnelle inverse
 - estAuteurDe est fonctionnelle inverse
- `:estAuteurDe a owl:InverseFunctionalProperty .`

Propriétés fonctionnelles

`:aPourAuteur a owl:FunctionalProperty .`

`:estAuteurDe owl:inverseOf :aPourAuteur .`

`:elmore :estAuteurDe :swag .`

`:swag :aPourAuteur :x .`

→ `:estAuteurDe a owl:InverseFunctionalProperty .`

→ `:swag :aPourAuteur :elmore .`

→ `:x owl:sameAs :elmore .`

Propriétés symétriques

- Si a est ami avec b, alors b est ami avec a
 `:aaa :friendOf :bbb .`
- `:friendOf a owl:SymmetricProperty ;`
 `rdfs:domain :C1 .`
 → `:friendOf a rdf:Property ;`
 `rdfs:range :C1 .`
 → `:bbb :friendOf :aaa .`

Propriétés transitives

- Les amis de mes amis sont mes amis

`:aaa :friendOf :bbb .`

`:bbb :friendOf :ccc .`

- `:friendOf` a `owl:SymmetricProperty` ,
`owl:TransitiveProperty` ;

`rdfs:domain :C1 .`

- **`→ :aaa :friendOf :ccc .`**
- **`→ :ccc :friendOf :aaa, :bbb .`**

Propriétés réflexives

- On est toujours son propre ami

`:aaa :friendOf :bbb .`

`:friendOf a owl:SymmetricProperty ,`

`owl:ReflexiveProperty .`

→ `:aaa :friendOf :aaa .`

→ `:bbb :friendOf :aaa, :bbb .`

Exemples

- `rdfs:subClassOf` est réflexive et transitive, mais pas symétrique
- `owl:sameAs` est réflexive, transitive et symétrique

Restrictions

- Permet de rajouter de la sémantique
 - à une propriété
 - À un type de données (littéraux)
- Une restriction est une classe (anonyme) à laquelle on ajoute des contraintes

Cardinalités

- Les créations artistiques ont un auteur
- Sans autre précision, une propriété a une cardinalité (0,n)
- `schema:author` associe un `schema:Person` à un `schema:CreativeWork`
`schema:author` `rdfs:domain` `schema:CreativeWork` ;
`rdfs:range` `schema:Person` .

Cardinalités

- Une œuvre collective peut avoir de 2 à 5 auteurs

```
:CollectiveWork rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:onProperty schema:author ;  
    owl:minCardinality 2 ;  
    owl:maxCardinality 5 ] .
```


Cardinalités

- Une œuvre à quatre mains a exactement 2 auteurs

```
:FourHandedWork rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:onProperty schema:author ;  
    owl:cardinality 2 ] .
```

Un fruit qui n'est pas orange...


Startpage.com

fruit qui n'est pas orange

Settings Support Mail

Web **Images** Videos News

Any size ▾ Any color ▾ Any type ▾



1 2 3 4 5 Next >

Un fruit qui n'est pas orange...

```
:fruit-orange rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:onProperty :aCouleur ;  
    owl:allValuesFrom [ oneOf (:orange) ] ] .  
  
:fruit-pas-orange rdfs:subClassOf [  
    owl:complementOf :fruit-orange ] .
```

Et encore...

- Beaucoup d'autres choses
- Ce qui est vu ici est représentatif
- Tout n'est pas disponible dans tous les profils

Pour quoi faire ?

- Une fois qu'on a une ontologie, il est possible (si le langage utilisé s'y prête) de faire des requêtes
- Triplestores
- SPARQL

Hypothèse du monde ouvert

- On a une base de faits (de triplets) :
 - T1
 - T2
 - T3
- Un nouveau triplet t4 peut être :
 - Soit forcément vrai (peut être déduit)
 - Soit forcément faux (entraîne une contradiction)
 - Soit *potentiellement* vrai (t4 enrichirait la base de faits)

Hypothèse du monde ouvert

```
:Cat rdfs:subClassOf :Animal .
```

```
:Dog rdfs:subClassOf :Animal .
```

```
:x a :Cat .
```

? :x a :Dog

→ potentiellement vrai

Hypothèse du monde ouvert

```
:Cat rdfs:subClassOf :Animal .
```

```
:Dog rdfs:subClassOf :Animal ;
```

```
    owl:disjointWith :Cat .
```

```
:x a :Cat .
```

```
? :x a :Dog
```

→ faux

Hypothèse du monde ouvert

```
:Cat rdfs:subClassOf :Animal .  
:Kitty rdfs:subClassOf :Animal ;  
      owl:equivalentClass :Cat .  
  
:x a :Cat .
```

? :x a :Kitty

→ vrai