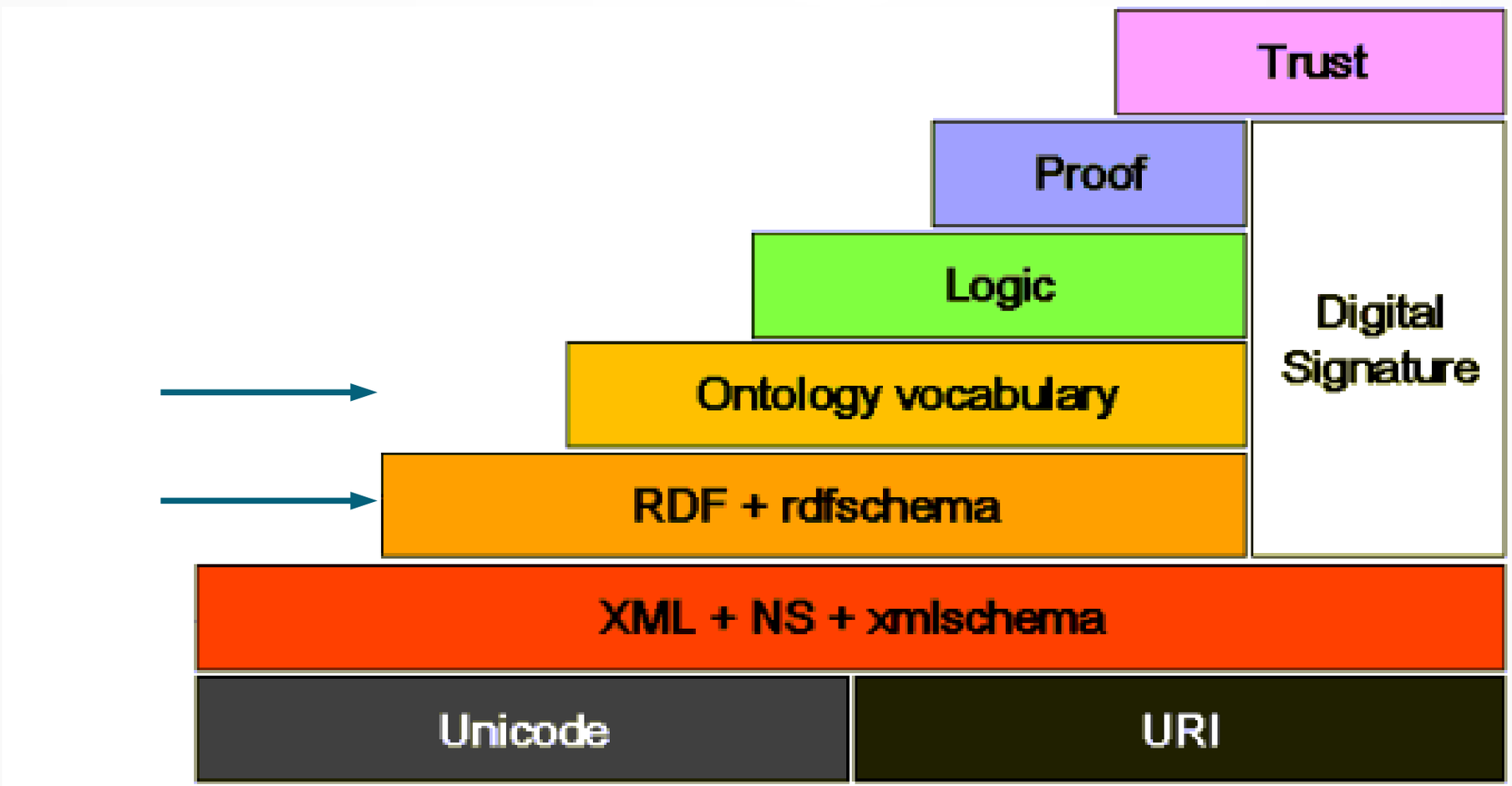
Le web sémantique

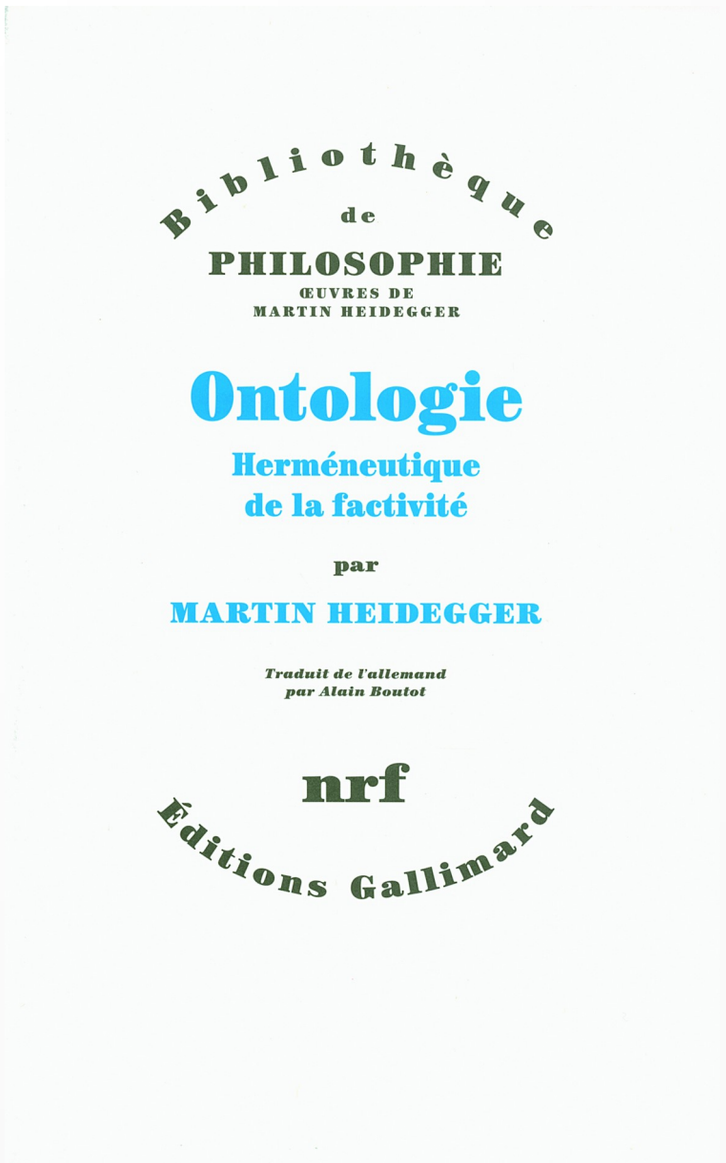
Vocabulaire RDF, RDF schémas, ontologies

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 règles pour déduire d'autres faits
- Comme un schéma de BD / un diagramme de classes, mais plus général

RDF : Resource Description Framework

- http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
- W3C en 1997, version 1.1 en 2014
- Associer des méta-informations à des ressources :
  - Page web
  - Item sur une page web
  - Document multimédia
  - …
- On peut créer ses propres annotations (extension du principe des microformats)
- On peut matérialiser les liens entre les ressources
- **Framework très généraliste (trop?)**

RDF : vocabulaire

- RDF, au-delà de la notion de triplets, définit des ressources
- On connaît surtout :
  - `rdf:type`
  - `rdf:List`
  - `rdf:first`
  - `rdf:rest`
  - `rdf:nil`

rdf:Property

- Est le type d'une propriété / prédicat
- C'est un type : commence par une majuscule
- Les propriétés commencent par une minuscule

```
rdf:type a rdf:Property .
rdf:first a rdf:Property .
rdf:rest  a rdf:Property .
```

rdf:Statement

- Est le type des triplets : *statement* = fait / affirmation
- Permet de faire de la *réification*

```
_:st1 a rdf:Statement .
```

- Comment dire quels sont les sujet, prédicat et objet de `_:st1` ?
- À quoi ça sert ?

rdf:subject, rdf:predicate, rdf:object

- Propriétés qui s'appliquent aux rdf:Statement (attention, ce qui suit est du Turtle)

```
ex:elmore a schema:Person .


_:st1 a rdf:Statement .
_:st1 rdf:subject ex:elmore .
_:st1 rdf:predicate rdf:type .
_:st1 rdf:object schema:Person .
```

Réification : à quoi ça sert ?

- Les triplets deviennent eux-mêmes des ressources
- Permet de décrire des méta-informations concernant les triplets eux-mêmes
- Différence entre :
  - Elmore est une personne
  - Fabien affirme qu'Elmore est une personne
- Incohérences, incertitudes : logiques modales

Réification

- Fabien affirme qu'Elmore est une personne

```
_:st1 a rdf:Statement .
_:st1 rdf:subject ex:elmore .
_:st1 rdf:predicate rdf:type .
_:st1 rdf:object schema:Person .
ex:fabien ex:asserts _:st1 .
```

Réification

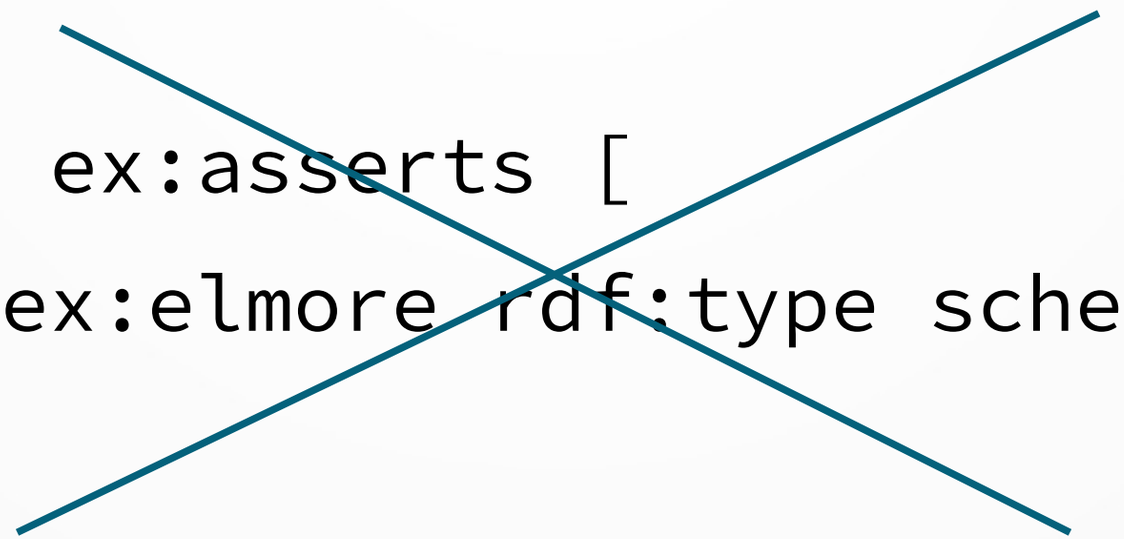
- Fabien affirme qu'Elmore est une personne

- ```
  ex:fabien ex:asserts [
              a rdf:Statement ;
              rdf:subject ex:elmore ;
              rdf:predicate rdf:type ;
              rdf:object schema:Person ] .
  ```
  ```

Réification

- NB : on ne pourrait pas écrire :

```
ex:fabien ex:asserts [
            ex:elmore rdf:type schema:Person
] .
```



- Parce qu'un nœud anonyme n'a pas de sujet

rdf:langString

- Est un type (malgré la minuscule)
- Décrit un littéral avec un tag de langue

rdf:XMLLiteral

- Représente un littéral de type fragment XML correct

```
ex:elmore ex:cv "<person><first-name>Elmore</first-name><last-
name>Leonard</last-name></person>"^^rdf:XMLLiteral .
```

Rappel concernant les littéraux

- En N-triples, obligation de mettre l'URI en entier
- En Turtle, on *peut* mettre le type en utilisant des préfixes

```
<http://example.org/totor> <http://example.org/age>
"23"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger> .

ex:totor ex:age 23 .

ex:totor ex:age "23"^^xsd:nonNegativeInteger .
```

rdf:HTML

- Représente un littéral de type fragment HTML correct

```
ex:elmore ex:cv "<div>Elmore Leonard a écrit <emph>Swag</emph>
</div>"^^rdf:HTML .
```

Les collections

- `rdf:Bag`
- `rdf:Seq`
- `rdf:Alt`
- Chaque élément de la collection est identifié via `rdf:_1`, `rdf:_2`, etc.

rdf:Bag, rdf:Seq, rdf:Alt

- `rdf:Bag`
  - Sac
  - Collection non ordonnée, avec doublons possibles
  - Un peu comme un ensemble (mais avec doublons)
- `rdf:Seq`
  - Séquence
  - Collection ordonnée, avec doublons possibles
- `rdf:Alt`
  - Alternatives
  - Disjonction
  - L'une des alternatives est choisie, par défaut la première

Exemple

- Elmore est l'auteur des livres *Swag*, *Killshot* et *Get Shorty*

```
ex:elmore ex:author [
        a rdf:Bag ;
        rdf:_1 ex:swag ;
        rdf:_2 ex:killshot ;
        rdf:_3 ex:get-shorty ] .
```

Exemple

- Elmore a écrit, dans l'ordre, *Swag*, puis *Killshot,* puis *Get Shorty*

```
ex:elmore ex:author [
        a rdf:Seq ;
        rdf:_1 ex:swag ;
        rdf:_2 ex:killshot ;
        rdf:_3 ex:get-shorty ] .
```

Exemple

- *Get Shorty* peut désigner, selon les cas, un film ou un roman

```
ex:get-shorty a [
        a rdf:Alt ;
        rdf:_1 schema:Movie ;
        rdf:_2 schema:Book ] .
```

rdf:value

- Décrit une valeur
- Utilisé dans des types structurés
- Pas de sémantique propre
- *Swag* est composé de 250 pages

```
ex:swag ex:consists-in [
        rdf:value 250,
        ex:unit ex:page ].
```

rdf:value

```
ex:clafoutis ex:hasIngredient
      [ ex:ingredient ex:egg ;
        rdf:value 6 ] ,
      [ ex:ingredient ex:flour ;
        rdf:value 250 ;
        ex:unit ex:gram ] .
```

RDF : déductions

- On peut déduire, uniquement en utilisant le vocabulaire rdf, des faits
- `ex:aaa ex:bbb ex:ccc → ex:bbb a rdf:Property`
- `ex:aaa rdf:subject ex:bbb → ex:aaa a rdf:Statement`
- `ex:aaa rdf:first ex:bbb → ex:aaa a rdf:List`
- Mais vocabulaire assez pauvre
- Trop pauvre pour être utilisé seul

RDF Schéma

- RDFS, RDF-S, RDF/S
- https://www.w3.org/TR/rdf-schema/
- 1998, version 1.1 en 2014
- Ajout de ressources permettant d'ajouter plus de sémantique
- Pouvoir descriptif / déductif plus élevé que RDF seul
- Souvent utilisé seul avec RDF

rdfs:Resource

- Une ressource
- Type très général : équivalent de Object en java

```
ex:aaa ex:bbb ex:ccc
```

→ `ex:bbb a rdf:Property`

rdfs:Resource

- Une ressource
- Type très général : équivalent de Object en java

```
ex:aaa ex:bbb ex:ccc
```

→ `ex:bbb a rdf:Property`

→ **`ex:bbb a rdfs:Resource`**

→ **`ex:aaa a rdfs:Resource`**

→ **`ex:ccc a rdfs:Resource`**

- NB : les propriétés sont des ressources

rdfs:Class

- Une classe, un type
- Un type, c’est tout ce qui peut être à droite de rdf:type

```
ex:aaa a ex:bbb
→ ex:aaa a rdfs:Resource
→ ex:bbb a rdfs:Resource
```

rdfs:Class

- Une classe, un type
- Un type, c'est tout ce qui peut être à droite de rdf:type

```
ex:aaa a ex:bbb
→ ex:aaa a rdfs:Resource
→ ex:bbb a rdfs:Resource
→ ex:bbb a rdfs:Class
```

- NB : les classes sont des ressources

rdfs:Literal

- Un littéral (un *t* en Anglais, deux en Français)
- NB
  - Les littéraux sont des ressources
  - Les `rdf:langString` sont des littéraux

rdfs:Datatype

- Un type auquel appartient un littéral

```
xsd:nonNegativeInteger a rdfs:Datatype
rdfs:Datatype a rdfs:Class
```

rdfs:Container

- La class qui regroupe les conteneurs RDF : Bag, Seq et Alt
- Mais pas List

rdfs:ContainerMembershipProperty

- La class qui regroupe les composantes des conteneurs RDF : `rdf:_1`, etc.

```
ex:elmore ex:author _:bag1 .

_:bag1 a rdf:Bag ;
       rdf:_1 ex:swag ;
       rdf:_2 ex:shorty .

→ _:bag1 a rdfs:Container, rdfs:Resource .
```

rdfs:subClassOf

- Indique qu'une classe est sous-classe d'une autre

```
schema:Author rdfs:subClassOf schema:Person .
ex:elmore a schema:Author .
→ ex:elmore a schema:Person .
```

- NB : toute classe est sous-classe d'elle-même
- Sémantique : a subClassOf b ↔ tous les a sont des b

rdfs:subClassOf

- Équivalences entre concepts

```
ex:aaa rdfs:subClassOf ex:bbb .
ex:bbb rdfs:subClassOf ex:aaa .
```

- Tous les aaa sont des bbb et tous les bbb sont des aaa
- aaa et bbb sont synonymes

rdfs:subPropertyOf

- Indique qu'une propriété est sous-propriété d'une autre
- Comme pour subClassOf, mais pour les propriétés
- Quand on est né dans un pays, on a déjà visité ce pays

```
ex:bornIn rdfs:subPropertyOf ex:knows .
ex:elmore ex:bornIn ex:USA .
→ ex:elmore ex:knows ex:USA .
```

- NB : toute propriété est sous-propriété d'elle-même
- Sémantique : a subPropertyOf b ↔ tous les a sont des b

rdfs:domain

- Indique ce qui est à gauche d'une propriété (le sujet)

```
rdfs:subClassOf a rdf:Property . # Par définition
rdfs:subClassOf rdfs:domain rdfs:Class . # Par définition
ex:aaa rdfs:subClassOf ex:bbb .
→ ex:aaa a rdfs:Class .

ex:aaa ex:bbb ex:ccc .
ex:bbb rdfs:domain ex:ddd .
ex:ddd rdfs:subClassOf ex:eee .
ex:bbb rdfs:subPropertyOf ex:fff .
→ ex:bbb a rdf:Property .
→ ex:aaa a ex:ddd , ex:eee ;
         ex:fff ex:ccc .
```

rdfs:range

- Indique ce qui est à droite d'une propriété (l'objet)

```
ex:aaa ex:bbb ex:ccc .
ex:bbb rdfs:domain ex:ddd .
ex:bbb rdfs:range ex:eee .
ex:ddd rdfs:subClassOf ex:eee .
ex:bbb rdfs:subPropertyOf ex:fff .
→ ex:bbb a rdf:Property .
→ ex:aaa a ex:ddd , ex:eee ;
         ex:fff ex:ccc .
→ ex:ccc a ex:eee . # Mais pas ex:ccc a ex:ddd
```

rdfs:domain, rdfs:range

- Exercice : décrivez les domain et range de toutes les propriétés vues jusqu'ici :
  - rdf:type,
  - rdf:first,
  - rdf:rest,
  - rdf:subjet,
  - rdf:predicate,
  - rdf:object,
  - rdfs:subClassOf,
  - Etc.

rdfs:domain, rdfs:range

- Exercice : décrivez les domain et range de toutes les propriétés vues jusqu'ici :
  - `rdf:type` :

```
rdf:type rdfs:domain rdfs:Resource ;
         rdfs:range rdfs:Class .
```

rdfs:label

- Une étiquette pour identifier en langage humain une ressource

```
ex:elmore rdfs:label "Elmore Leonard" .


rdfs:label a rdf:Property ;
           rdfs:domain rdfs:Resource ;
           rdfs:range rdfs:Literal .
```

rdfs:comment

- Un commentaire pour décrire en langage humain une ressource
- Idéal pour décrire une classe ou une propriété

```
ex:author a rdf:Property ;
          rdfs:domain ex:Book ;
          rdfs:range ex:Person ;
          rdfs:comment "Indique la personne étant l’auteur
du livre"@fr ;

rdfs:comment rdfs:domain rdfs:Resource ;
             rdfs:range rdfs:Literal .
```

rdfs:member

- Utilisé dans les collections
- Super-propriété de `rdf:_1, rdf:_2`, etc.

```
rdfs:member a rdf:Property .
rdf:_1 rdfs:subPropertyOf rdfs:member .
rdf:_2 rdfs:subPropertyOf rdfs:member .
```

```
_:bag1 rdf:_1 ex:truc → _bag1 rdfs:member ex:truc
```

rdfs:seeAlso

- Indique que l'objet contient des renseignements complémentaires sur le sujet

```
ex:elmore rdfs:seeAlso
<https://en.wikipedia.org/wiki/Elmore_Leonard> .
```

rdfs:isDefinedBy

- Indique que l'objet définit le sujet
- Sémantique plus précise que seeAlso

```
rdfs:isDefinedBy rdfs:subPropertyOf rdfs:seeAlso .
```

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

Monde ouvert

```
:Cat rdfs:subClassOf :Animal .
:Dog rdfs:subClassOf :Animal .
:barksAt a rdf:Property ;
          rdfs:domain :Dog ;
          rdfs:range rdfs:Resource .
:freyja a :Cat .
:freyja :barksAt [] .

→ :freyja a :Dog .
```

OWL : Web Ontology Language

- Langage beaucoup plus expressif
- Mais attention à la complexité algorithmique