Le web sémantique

Le web avant google...

- Pas facile d'y retrouver des informations sans un bon moteur de recherche.
- Synonymes, fautes de frappe, contenu optimisé pour remonter dans les résultats, ...

Le web après google...

- Toutes les villes de plus de 10 000 habitants à moins de 75 km de Lille.
- Toutes les entreprises qui recrutent des ingénieurs en informatique à moins de 30 km de chez moi.
- Nom d'un fruit qui n'est pas orange.

Le web sémantique

- 2001, idée popularisée par Tim Berners-Lee.
- Ajouter du sens aux contenus web: description formelle :
  - Page qui traite d'un sujet
  - Événement
  - Coordonnées
  - …
- Recherche de contenus plus simple et plus précise.
- Mais nécessite un travail en amont par les créateurs de contenu.

Exemple 1 : les fruits

- La pomme est un fruit
- La pomme peut être de couleur rouge
- La pomme peut être de couleur verte
- La pomme peut être de couleur jaune
- La pomme pousse en Europe
- La pomme pousse en Amérique
- …

Exemple 1 : les fruits

- Un légume-fruit est un légume
- Un légume-fruit est un fruit
- Solanum lycopersicum est un légume-fruit
- Solanum lycopersicum est synonyme de tomate
- La tomate peut être de couleur rouge
- La tomate peut être de couleur noire
- La tomate peut être de couleur jaune
- ...

Exemple 1 : les fruits

- Le poivron est un légume-fruit
- Le poivron peut être de couleur rouge
- Le poivron peut être de couleur verte
- Le poivron peut être de couleur orange
- …
- Le poivron est-il un fruit qui n’est pas orange ?

Microformats

- HTML décrit la structure d'un document
  - Ceci est un titre
  - Ceci est un paragraphe
  - Ceci est un lien
  - Ceci est une image
  - Ceci est important
- Pas la mise en forme (CSS)
- Pas non plus la sémantique du contenu

Microformats

- HTML ne décrit pas la sémantique :
  - Ceci est une biographie
  - Ceci est le centre d'intérêt d'une personne
  - Ceci est la photo de la personne
  - Ceci est l'adresse de la personne
  - Ceci est une date
  - …
- Ajout de méta-informations dans le code HTML

Microformats

- Beaucoup de microformats différents
- Certains reconnus et traités par les navigateurs
- Certains compris par les moteurs de recherche
- Certains compris par les réseaux sociaux

Microformat : hCalendar

- Pour signaler des événements
- Avant :

```
<p>
   Le cours de XML du lundi 19 novembre aura lieu de
   8h15 à 11h15 en salle G309 et promet d'être
   passionnant.
</p>
```

Microformat : hCalendar

Après :

```
<p class='vevent'>
   Le <span class='summary'>cours de XML</span> du lundi 19
   novembre aura lieu de
   <abbr class='dtstart' title='2018-11-
   19T08:15:00+01:00'>8h15</abbr>
   à
   <abbr class='dtend' title='2018-11-
   19T11:15:00+01:00'>11h15</abbr>
   en <span class='location'>salle G309</span> et promet
   d'être passionnant.
</p>
```

Microformat : hCard

- Carte de visite (infos de contact)
- Avant :

```
<p>
  Fabien Delorme<br/>
  Faculté Jean Perrin<br/>
  Rue Jean Souvraz, Lens<br/>
  03 21 79 17 50<br/>
  <a href='https://www.cril.fr/~delorme'>page perso</
  a>
</p>
```

Microformat : hCard

- Après :

```
<p class='vcard'>
  <span class='fn'>Fabien Delorme</span><br/>
  <span class='org'>Faculté Jean Perrin</span><br/>
  <span class='street-address'>Rue Jean
  Souvraz</span>,
  <span class='locality'>Lens</span><br/>
  <span class='tel'>03 21 79 17 50</span><br/>
  <a class='url'
  href='https://www.cril.fr/~delorme'>page perso</a>
</p>
```

Microformat : OpenGraph

- Facebook : ogp.me
- Faciliter le partage sur les réseaux sociaux

```
<html prefix="og: http://ogp.me/ns#">
   <head>
      <title>The Rock (1996)</title>
      <meta property="og:title" content="The Rock"/>
      <meta property="og:type" content="video.movie"/>
      <meta property="og:url" content="http://www.imdb.com/title/tt0117500"/>
      <meta property="og:image"
      content="http://ia.media-imdb.com/images/rock.jpg"/>
   </head>

   ...
```

Microformats

- Avantages :
  - Très simple à mettre en place
    - Un document XML contient des infos sémantiques : facile de passer d'un doc XML à un doc HTML enrichi sémantiquement
  - Très simple à traiter

Microformats et XML

- XML :

```
<authors>
  <author name='Fabien Delorme'>
    <organization>Faculté Jean Perrin</organization>
    <address>
      <street>Rue Jean Souvraz</street>
      <city>Lens</city>
    </address>
    <phone val='03 21 79 17 50'/>
   -<webpage url='https://www.cril.fr/~delorme'/>
    <articles>
      …
    </article>
  </author>
  ...
</authors>
```

Microformats et XML

- La feuille XSLT peut conserver les méta-informations :

```
<p class='vcard'>
  <span class='fn'>Fabien Delorme</span><br/>
  <span class='org'>Faculté Jean Perrin</span><br/>
  <span class='street-address'>Rue Jean
  Souvraz</span>,
  <span class='locality'>Lens</span><br/>
  <span class='tel'>03 21 79 17 50</span><br/>
  <a class='url'
  href='http://www.cril.fr/~delorme'>page perso</a>
</p>
```

Microformats

- Inconvénients :
  - Méta-informations sur la ressource actuelle uniquement
  - Quid des liens entre ressources ?
    - On sait que le cours a lieu à la « fac Jean Perrin »
    - C'est quoi la « fac Jean Perrin » comme type de lieu ?
    - Je veux connaître la liste des concerts qui se dérouleront cette semaine près de chez moi : je dois identifier les salles de concert
    - La « fac Jean Perrin » est une faculté : OK, mais c'est quoi une faculté ?
    - Liens entre ces ressources : ontologie

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 et la notion de triplet

- Toutes les ressources peuvent être décrites formellement sous forme de triplets
- **sujet** *verbe* complément
- **sujet** *prédicat/propriété* objet
- **Totor le castor** *s'intéresse à* Web sémantique
- **:totor** *foaf:interest* w3c:semWeb
- L'ensemble des triplets forme un graphe
- Toutes les ressources peuvent être sujet, prédicat ou objet

Syntaxes RDF

- Plusieurs syntaxes !
  - RDF XML
  - Notation 3
  - Turtle (sous-ensemble de Notation 3)
  - N-triples (sous-ensemble de turtle)
  - Etc.

N-triples

- `sujet prédicat objet .`
- `tomate est-un légume-fruit .`
- `légume-fruit est-un légume .`
- `légume-fruit est-un fruit .`
- `tomate couleur rouge .`
- `tomate couleur noir .`
- …

N-triples

```
<uri-sujet> <uri-prédicat> <uri-objet> .

<http://www.example.org/#totor>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Person> .

<http://www.example.org/#totor>
<http://xmlns.com/foaf/0.1/name>
"Totor le Castor" .
```

N-triples

- En RDF, les prédicats/propriétés sont des ressources aussi :

```
<http://xmlns.com/foaf/0.1/name>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/1999/02/22-rdf-syntax-
ns#property> .
```

N-triples : les littéraux

```
<http//www.example.org/#totor>
<http://xmlns.com/foaf/0.1/age>
"23"^^<http://www.w3.org/2001/XMLSchema#integer> .

<http//www.example.org/#totor>
<http://xmlns.com/foaf/0.1/name>
"Totor the Castor"@en .
```

N-triples : blank nodes

- Noeuds vides : nœuds anonymes
- Pas associés à une ressource identifiée via une URI

```
<http://www.example.org/#totor>
<http://www.example.org/#lives-with>
_:ada.

_:ada <http://xmlns.com/foaf/0.1/name> ‘Ada’.
```

-
```
_:ada
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.example.org/#Cat>.
```

N-triples

- Avantages :
  - Très simple à écrire
  - Très simple à parser
  - Grammaire de quelques lignes
- Inconvénients :
  - Peut être très verbeux

Turtle

- Possibilité de créer des préfixes
- Virgules et point-virgules pour éviter les répétitions

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-
syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://www.example.org/> .
ex:totor rdf:type foaf:Person ;
         foaf:name "Totor le Castor" ,
                   "Totor the Castor"@en .
foaf:name rdf:type rdf:property .
```

Turtle

- Entiers, décimaux, réels
- Listes
- Booléens
- …

Turtle

- Avantages :
  - Très « human-friendly » : facile à lire, comprendre et écrire pour un humain
  - Idéale pour écrire des descriptions à la main
  - Relativement facile à parser
  - Plus compact que n-triples
- Inconvénients :
  - Plus difficile à parser que n-triples
  - Pas une syntaxe XML

RDF-XML

```
<rdf:RDF xmlns:rdf='...'>
  <rdf:Description rdf:about='sujet'>
    <predicat rdf :resource='objet'>
    <predicat>objet littéral</prédicat>
  </rdf:Description>
</rdf:RDF>
```

RDF-XML

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-
  ns#'
  xmlns:ex='http://www.example.org'
  xmlns:foaf='http://xmlns.com/foaf/0.1/'>
  <rdf:Description rdf:about='ex:totor'>
    <rdf:type rdf:resource='foaf:Person'/>
    <foaf:name>Totor le Castor</foaf:name>
    <foaf:age>23</foaf:age>
  </rdf:Description>
</rdf:RDF>
```

RDF-XML

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:ex='http://www.example.org'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  xmlns:foaf='http://xmlns.com/foaf/0.1/'>
  <rdf:Description rdf:about='ex:totor'>
    <rdf:type rdf:resource='foaf:Person'/>
    <foaf:name>Totor le Castor</foaf:name>
    <foaf:name xml:lang='en'>Totor the Castor</foaf:name>
    <foaf:age rdf:datatype='xs:integer'>23</foaf:age>
  </rdf:Description>
</rdf:RDF>
```

RDF-XML

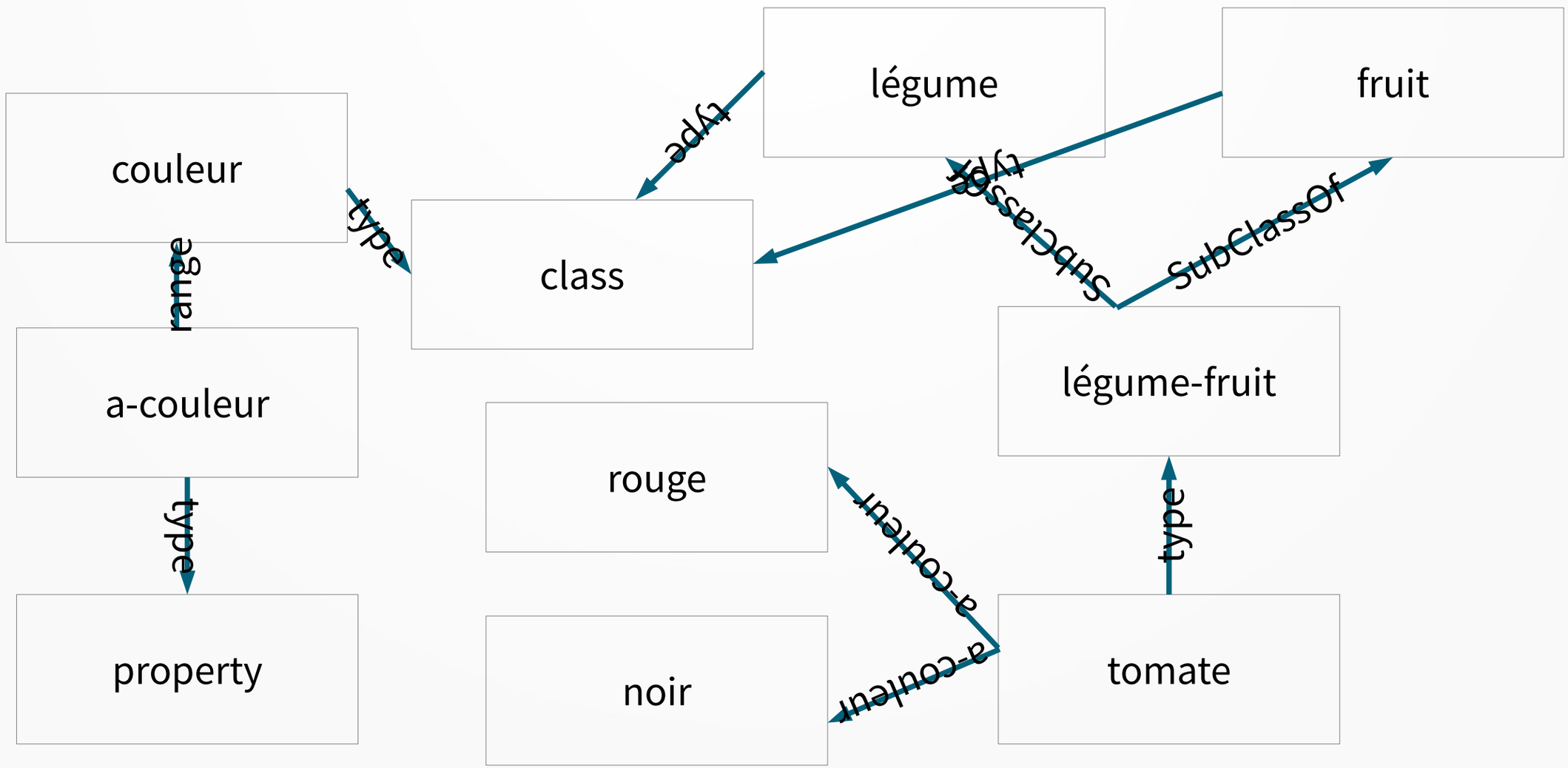
```
<rdf:Description
  rdf:about='sujet'
  predicat='objet littéral'/>

<rdf:Description
  rdf:about='ex:totor'
  foaf:name='Totor le Castor'
  foaf:age='23'/>
```

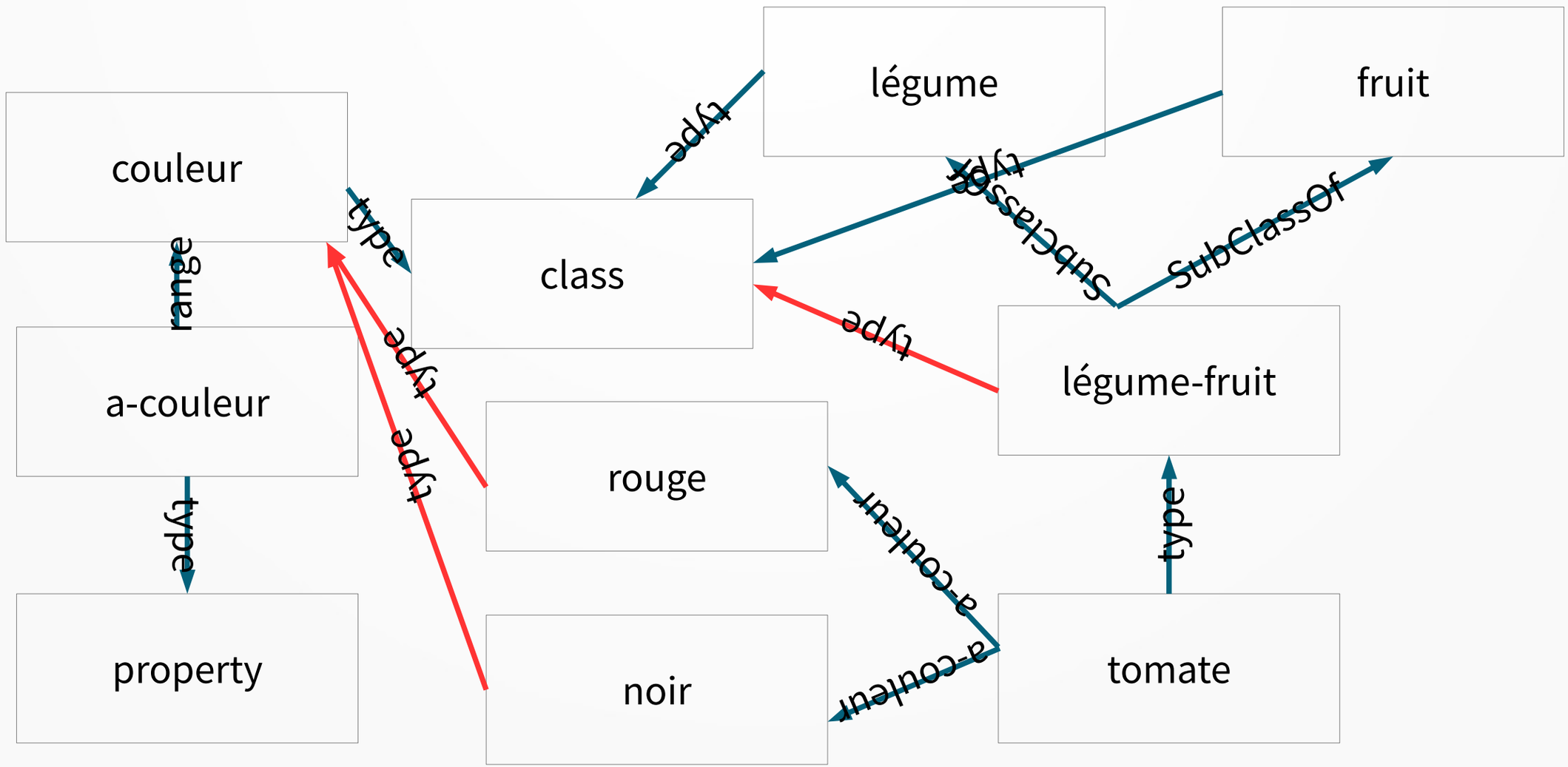
Les triplets forment un graphe

```
légume type class .
fruit type class .
légume-fruit subClassOf légume .
légume-fruit subClassOf fruit .
tomate type légume-fruit .
a-couleur type property .
a-couleur range couleur .
couleur type class .
tomate a-couleur rouge .
tomate a-couleur noir .
```

Graphe associé



Déductions possibles



Fun with triples : restrictions

Un légume vert est un légume dont la couleur est forcément le vert

```
légume-vert type class ;
            subClassOf légume ,
                       [ type Restriction ;
                         onProperty aCouleur ;
                         allValues vert ].
```

Réécriture en triplets

```
légume-vert subClassOf légume ,
                       [ type Restriction ;
                         onProperty aCouleur ;
                         allValues vert ].

légume-vert subClassOf légume .
légume-vert subClassOf _restr .
_restr type Restriction .
_restr onProperty aCouleur .
_restr allValues vert .
```

La vraie version…

```
légume-vert type Class ;
            subClassOf légume ,
                              [type Restriction ;
                               onProperty aCouleur ;
                               allValuesFrom [type Class ;
                                              oneOf(vert)
                                             ]
                              ] .
```

La vraie version...

```
_restr type Restriction .
_restr onProperty aCouleur .
_restr allValuesFrom _vals .
_vals type Class .
_vals oneOf _lst .
_lst first vert .
_lst rest nil .
```

RDF et les listes : first, rest et nil

- Une liste en Turtle :

```
x oneOf (a, b, c) .
```

- Une liste en n-triples :

```
x oneOf _lst .
_lst first a .
_lst rest _lst2 .
_lst2 first b .
_lst2 rest _lst3 .
_lst3 first c .
_lst3 rest nil .
```

Quelques grammaires RDF...

- RDF est trop généraliste
- Comment raisonner sur un ensemble de triplets RDF ?
- Comment connaître la sémantique associée à un prédicat ?
- Il faut un vocabulaire commun
  - RDF Schema
  - Friend of a friend
  - Dublin core
  - OWL
  - …

RDFS : RDF Schema

- http://www.w3.org/TR/rdf-schema/
- Créé par le W3C en 1998
- Version finale en 2004
- Grammaire pour le typage basique des ressources
- Permet de raisonner (un peu) sur les documents RDF
- Permet de faire des requêtes via SPARQL

RDFS : RDF Schema

```
ex:Cat rdf:type rdfs:Class ;
       rdfs:subClassOf ex:Animal .
ex:eat rdf:type ex:Activity .
ex:sleep rdf:type ex:Activity .
ex:Activity rdf:type rdfs:Class .
ex:likes rdf:type rdf:property ;
         rdfs:domain ex:Cat ;
         rdfs:range ex:Activity .
ex:ada rdf:type ex:Cat ;
       ex:likes ex:sleep ,
                ex:eat .
ex:greypuss ex:likes ex:kill-birds .
```

RDFS : quelques types

- rdfs:Resource
- rdfs:Class
- rdfs:Literal
- rdfs:Datatype

RDFS : quelques prédicats

- rdfs:subClassOf
- rdfs:subpropertyOf
- rdfs:domain
- rdfs:range
- rdfs:label
- rdfs:comment

FOAF : Friend of a friend

- http://xmlns.com/foaf/spec/
- Dernière version : 2014
- Description d'infos sur les personnes :
  - État civil,
  - Centres d'intérêt,
  - Liens entre personnes,
  - …
- Utilise d'autres grammaires : Dublin Core, RDFs, OWL

FOAF

- Types `Person`, `Document`, `Organization`
- Type `Agent` (superclasse de `Person`, `Document`, `Organization`)
- Type `Group` (ensemble d'`Agents`)

FOAF : Person

- Quelques propriétés des `Persons` :
  - firstName → string
  - lastName → string
  - surname → string
  - phone → string
  - img → URL
  - interest → Thing
  - currentProject → Thing
  - knows → Person

FOAF : exemple depuis DBLP

- ex:fabien-delorme foaf:firstName ‘Fabien’.
- ex:fabien-delorme foaf:lastName ‘Delorme’.
- ex:fabien-delorme foaf:knows ex:sylvain-lagrue.
- ex:fabien-delorme foaf:knows ex:nathalie-chetcuti
- ex:sylvain-lagrue foaf:firstName ‘Sylvain’.
- Etc.
- On peut connaître toutes les connaissances d’une personne :
  - Directement
  - Indirectement (niveau 2, 3, n)

DC : Dublin Core

- http://dublincore.org/
- Initiative gouvernementale des USA
- 1995
- Associer des métadonnées aux ressources
- Assurer l'interopérabilité entre systèmes d'information hétéroclites
- Plus vieux que XML, RDF, web sémantique

DC : Dublin Core

- Informations sur une ressource :
  - Title
  - Creator
  - Subject
  - Description
  - Publisher
  - Date
  - Langage
  - Identifier
  - Rights
  - …

DC : Dublin Core

- Format quasi-officiel
- Simple à mettre en œuvre
- Utilisé par beaucoup d’organismes officiels

OWL : Web Ontology Language

- Aller encore plus loin que RDFS
- Ontologies : conceptualisation d'un domaine de connaissance
- Plusieurs niveaux :
  - 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

OWL : quelques possibilités

- Équivalence, classes disjointes
- Propriété fonctionnelle
- Cardinalités
- Négation, intersection, union
- Transitivité, relation inverse
- Etc…

Les concepts / classes

- Catégorise un ensemble d'individus
- Taxonomies : sous-classes
  - Tous les mammifères sont des animaux
- Classes équivalentes
  - Une voiture et une automobile, c'est la même chose
  - A = B
  - → A est-un B, B est-un A
- Classes disjointes
  - Un chat et un chien, ce n'est jamais la même chose
  - Un chat et un animal domestique, ça peut être la même chose

Hypothèse du monde ouvert

- Habituellement, ce qui n'est pas décrit est forcément faux
  - UML : A n'est pas sous-classe de B ? Alors les A ne sont jamais des B
- Ontologies (en général) : ce qui n'est pas décrit peut être vrai ou faux
  - OWL : A n'est pas sous-classe de B ? Peut-être que certains A sont des B

Négation

- Les éléments stables sont des éléments non-radioactifs
  - Stable = non(Radioactif)
- Peut introduire des incohérences
  - A = non(B)
  - A est-un C
  - C est-un B

Intersections de concepts

- Les légumes fruits sont à la fois des légumes et des fruits
- Vrai pour les légumes → vrai pour les légumes-fruits
- Vrai pour les fruits → vrai pour les légumes-fruits
  - A = B et C
  - B = non(D)
  - → A = non(D)

Union de concepts

- Un joueur est un humain ou une IA
  - A = (B ou C)
  - B = non(D)
  - On ne peut pas déduire que A = non(D)
  - C = D
  - Pas d’incohérence
  - Attention : complexité élevée !

Les individus

- Instances de classes
  - Tomate est-un Légume-Fruit
  - Légume-Fruit est-un (Légume et Fruit)
  - Fruit est-un Truc-Coloré
  - → Tomate est-un Truc-Coloré
  - Fruit est-un non(Tubercule)
  - Patate est-un Tubercule
  - → Patate est-un non(Légume-Fruit)

Les relations / propriétés

- Associent deux individus
  - Eats(ada, randomMouse)
- Sont typées : domain et range
  - domain(Eats) = Animal
  - range(Eats) = Living-Thing
  - Animal est-un Living-Thing
  - → Animal(ada), Living-Thing(randomMouse)

Les relations / propriétés

- Eats-meat est-un Eats
  - Eats-meat(ada, someMouse)
  - → Eats(ada, someMouse), Animal(Ada), Living-Thing(someMouse)
- range(Eats-meat) = Animal
  - → Animal(someMouse)
- Eats-vegetable est-un Eats
- range(Eats-vegetable) = not(Animal)
- Ominvore = exists(Eats-meat) and exists(Eats-vegetable)
- Carnivore = exists(Eats-meat) and not exists(Eats-vegetable)
- Attention à la complexité !

Les relations : arités

- Eats-healthy = exists(>=5)(Eats-vegetable)
- Attention à la complexité !
- Relation fonctionnelle
  - Il n'y a qu'une seule instance de la relation pour un individu donné
  - Une personne n'habite qu'à un seul endroit : habite-à est fonctionnelle
  - Une personne peut travailler à plusieurs endroits : travaille-à n'est pas fonctionnelle

Les relations

- Transitivité
  - si a > b et b > c alors a > c
  - Si a est-chef-de b et b est-chef-de c alors a est-chef-de c
- Relation inverse
  - si a > b alors a < b
  - Si a est-chef-de b alors b est-subalterne-de a

Un langage pour chaque usage

- OWL complet : très riche
- Complexité rédhibitoire : inférence et requêtes complexes : indécidable
- Utile pour formaliser précisément un domaine de connaissance complexe

Profils OWL : OWL 2 EL

- Très efficace pour des ontologies très riches
- Déterminer en temps polynomial si un individu est instance d'un concept ou si une classe est sous-classe d'une autre
- Très limité en expressivité
  - Pas de négation
  - Pas d'union
  - Pas de cardinalité
  - Pas de transitivité
  - Etc.

Profils OWL : OWL 2 QL

- Plus complet qu'OWL EL
- Suffisamment riche pour décrire un modèle UML / un schéma de BD / un schéma XML
- Permet de faire les mêmes requêtes que sur une BD, en temps polynomial
  - Pas de cardinalités
  - Pas de transitivité
  - Pas d'union
  - Etc.

Profils OWL : OWL 2 RL

- Plus riche que les précédents, mais avec quelques restrictions
- Permet de raisonner, mais pas d'algo polynomiaux
- Utilisable uniquement sur de petites ontologies

Ontologies au-delà d'OWL

- Pas obligé d'utiliser un des profils
- On peut faire son langage à la carte
- Certaines constructions sont coûteuses
- Pas obligé d'utiliser OWL !
- RDFS seul est déjà très puissant

Triplestores & SPARQL

Une DTD en RDF

```
<!ELEMENT dblp (article|inproceedings)*>
<!ATTLIST dblp mdate CDATA #IMPLIED >

<!ENTITY % field "author|editor|title|year|">

<!ELEMENT article (%field;)*>
<!ELEMENT inproceedings (%field;)*>
```

Une DTD en RDF

```
Dblp type Class .
hasArticle type property ;
           domain Dblp ;
           range Article .
hasInproceedings type property ;
                 domain Dblp ;
                 range Inproceedings.
Field type Class.
Author subClassOf Field.
Editor subClassOf Field.
Title subClassOf Field.
Year subClassOf Field.
hasField type property ;
         domain [ oneOf (Article, Inproceedings) ] ;
         range Field .
```

Triplestores

- On peut transformer toute la base DBLP en triplets RDF !
- BD pour stocker les triplets : triplestore
  - Implémentations « from scratch »
  - Implémentation par-dessus SGBD SQL
  - Implémentations par-dessus SGBD NoSQL

SPARQL

- Langage de requêtes adapté aux triplets
- « SPARQL Protocol and RDF Query Language »
- Recommandation W3C (2008)
- (source des exemples : wikipédia)

SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
       ?email
WHERE
  {
    ?person  a           foaf:Person .
    ?person  foaf:name   ?name .
    ?person  foaf:mbox   ?email .
  }
```

SPARQL

```
PREFIX ex: <http://example.com/exampleOntology#>
SELECT ?capital
       ?country
WHERE
  {
    ?x  ex:cityname        ?capital   ;
        ex:isCapitalOf     ?y         .
    ?y  ex:countryname     ?country   ;
        ex:isInContinent   ex:Africa  .
  }
```

SPARQL : DBLP

```
SELECT ?y
WHERE
   {
      ex:totor foaf:knows ?x .
      ?x foaf:knows ?y .
   }
```

SPARQL : DBLP

```
SELECT ?y
WHERE
   {
      ex:totor foaf:knows ?x .
      ?x foaf:knows ?y .
      FILTER (?y != ex:totor) .
   }
```

SPARQL : DBLP

```
SELECT ?y
WHERE
  {
    { ex:totor foaf:knows ?x .
      ?x foaf:knows ?y .
      FILTER (?y != ex:totor) .
    }
    UNION
    { ex:totor foaf:knows ?x1 .
      ?x1 foaf:knows ?x2 .
      ?x2 foaf:knows ?y .
      FILTER (?y != ex:totor) .
    }
  }
```