Introduction au Web Sémantique

INF4230 – Amal Zouaq

Relation WS/ IA (Selon Berners-Lee)

- Le WS n'est pas de l'IA et l'IA n'est pas le WS
- L'IA est un domaine, le WS est un projet
- Le WS emprunte beaucoup à l'IA
- Le WS pourrait être un grand terrain de jeu pour l'IA
- Les projets IA devrait utiliser le WS pour inter-opérer.

Introduction

- WWW : succès fondé sur sa simplicité, MAIS !
- Développé pour des lecteurs humains
  - Les données actuelles sont principalement organisées et structurées pour être simples à transmettre et être présentées à des humains
- Or, Internet est de plus en plus utilisé par des machines – moteurs de recherche, agents, robots, etc.

Introduction

- Beaucoup de tâches nécessitent de combiner des données sur le Web:
  - Les hôtels et les informations de voyages peuvent être trouvés sur des sites différents
  - On peut effectuer des recherches dans différentes librairies digitales
  - etc.
- Les humains combinent ces informations très facilement
- … Même si différentes terminologies sont utilisées!

Introduction

- MAIS: les machines sont ignorantes!
  - L'information partielle n'est pas vraiment utilisable
  - Il leur est difficile d'associer un sens, à une image par exemple
  - Il leur est difficile d'effectuer des analogies automatiquement
  - Il leur est difficile de combiner de l'information automatiquement
  - Est-ce que **<foo:creator>** est équivalent à **<bar:author> ?**
  - Comment combiner différentes hiérarchies XML?
  - …

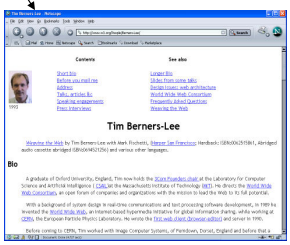
Exemple: Réservation automatique de vol

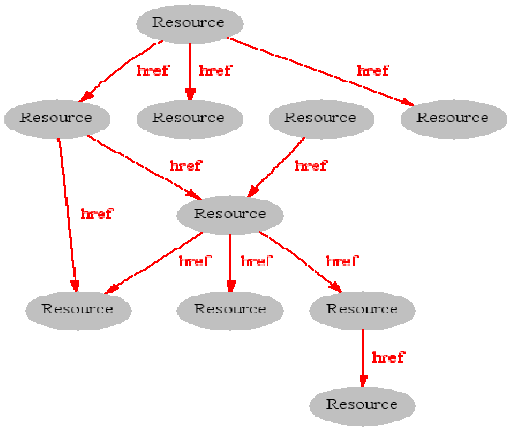
- Le système de réservation automatique
  - Connaît vos préférences
  - Construit une BC en utilisant votre passé (brrr!)
  - Peut combiner la connaissance locale avec des services distants:
    - Préférences de compagnies de vol
    - Préférences alimentaires
    - etc.
- Il communique avec de l'information distante (i.e., sur le Web!)

Problèmes de la recherche d'information sur internet

- Entrave la recherche, l'extraction, la maintenance et la génération d'information
- Actuellement, pas d'accès réel au contenu des documents
- Contenu et information non accessibles ni interprétables par des machines
- Pas possible de composer dynamiquement des documents cohérents et adaptés aux utilisateurs

Ou en sommes-nous aujourd'hui : le Web syntaxique (*the Syntactic Web*)





Le Web Syntaxique est…

- Une bibliothèque hypermédia numérique
  - Une bibliothèque de documents appelés (pages web) interconnectés par un hypermédia de liens
- Une base de données, une plate-forme d'application
  - Un portail commun aux applications, accessibles par le biais de pages Web, et présentant leurs résultats sous forme de pages Web
- Une plate-forme pour le multimédia
  - Radio, etc.,
- Un schéma de nommage (A naming scheme)
  - Identité unique pour les documents
- Un endroit où les ordinateurs se chargent de la présentation (facile) et où les gens effectuent les liens et l'interprétation (difficile)
- Pourquoi ne pas faire en sorte que les ordinateurs se chargent de la partie difficile ??!

[Goble 03]

Impossible (?) en utilisant le web syntaxique…

- Effectuer des requêtes complexes requérant de la connaissance (background knowledge)
  - Trouver de l'information sur "des animaux qui utilisent des sonars mais qui ne sont ni des chauves-souris ni des dauphins"
- Trouver de l'information dans des entrepôts de données
  - Recherche de voyages
  - Prix des biens et services
  - …
- Trouver et utiliser des "services web"
- Déléguer des tâches complexes à des agents
  - Réserver un séjour pour le prochain we dans un endroit chaud, pas trop loin et où la langue parlée est le français ou l'anglais

Quel est le Problème?

Considérons une page web typique:

Les annotations consistent en:

Le rendu des informations (par exemple, la taille de police et couleur)

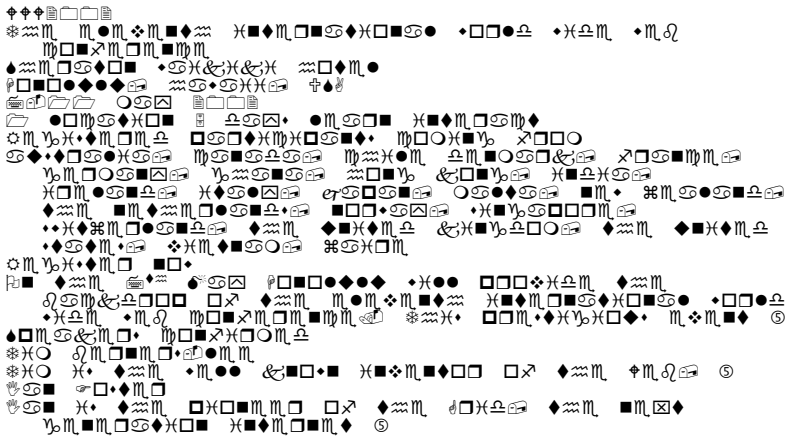
Hyperliens vers des contenus

Le contenu sémantique est accessible à l'homme mais pas (facilement) à des ordinateurs …

Voici l'information que les humains peuvent voir:

WWW2002
The eleventh international world wide web conference
Sheraton waikiki hotel
Honolulu, hawaii, USA
7-11 may 2002
1 location 5 days learn interact
Registered participants coming from
australia, canada, chile denmark, france, germany, ghana, hong kong, india, ireland, italy, japan, malta, new zealand, the netherlands, norway, singapore, switzerland, the united kingdom, the united states, vietnam, zaire
Register now
On the 7th May Honolulu will provide the backdrop of the eleventh international world wide web conference. This prestigious event …
Speakers confirmed
Tim berners-lee
Tim is the well known inventor of the Web, …
Ian Foster
Ian is the pioneer of the Grid, the next generation internet …

Voici l'information que les ordinateurs peuvent voir:



Définition

- "The Semantic Web is an **extension of the current** web in which information is given **well-defined meaning, better enabling computers and people to** work in cooperation."

-- Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, *Scientific American, Mai 2001.*

- Les données doivent être encodées de tel sorte que les contenus – sens et sémantique - du web soient plus « compréhensibles » pour les machines, ou en d'autres termes par des algorithmes.

Objectifs du WS

- Interopérabilité des données à travers des applications et des organisations (pour IT)
- Un ensemble de normes interopérables pour l'échange de connaissances
- Une architecture pour l'interconnexion des communautés et des vocabulaires

Le web actuel

- Ensemble de documents
- Basé essentiellement sur HTML
- Recherche par mots clé
- Utilisable par l'humain

Le web sémantique

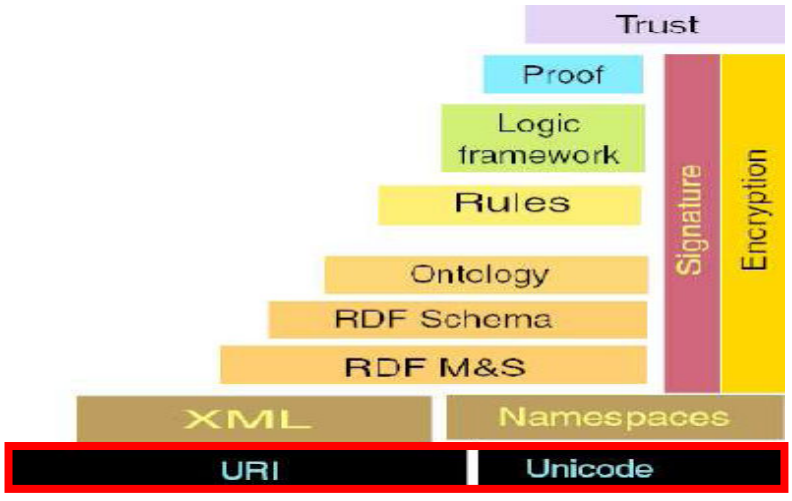
- Ensemble de connaissances
- Basé sur XML, RDF(S), OWL
- Recherche par concepts
- Utilisable par la machine

http://www.cours.polymtl.ca/inf4215/documentation/tutorielWS.pdf

Quelques mythes à propos du WS

- *M1 : Le WS permet de constituer une seule grande ontologie*
  Le WS est un ensemble d'ontologies interconnectées….
- *M2 : Toutes les ontologies du Web sémantique doivent être consistantes*
  Seulement les parties à utiliser ensemble!!!!!!!

Les couches du web sémantique



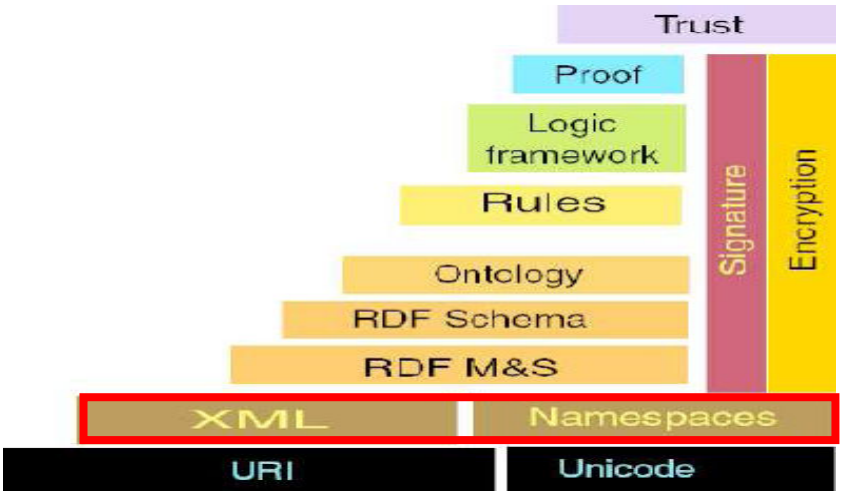
URI (Uniform Resource identifier)

- URI : Uniform Resource Identifier
- Identifie une entité à laquelle on réfère sur le web.
  - Chacun peut en créer – décentralisé
- Cette entité n'est pas nécessairement une ressource disponible sur le web.
- Protocoles: ftp, http, IMAP, SMTP, etc. (normalisée)
- Exemple d'URI: http://en.wikipedia.org/wiki/Uniform_Resource_Identifier

Unicode

- Unicode est un système d'encodage de caractères universel, mis à jour par le Consortium Unicode.
- Le standard Unicode est un système conçu pour appuyer les échanges, le traitement et l'affichage des textes écrits de la diversité des langues et des disciplines techniques
- Unicode spécifie un nombre unique pour chaque caractère, quels que soient la plate-forme, le logiciel et la langue utilisés.
- Unicode traduit chaque caractère en 16 bits et peut donc prendre en compte plus de 65.000 caractères uniques, et traiter informatiquement tous les systèmes d'écriture de la planète.

Les couches du web sémantique



Espace de nom (Namespace)

- En général, un espace de nom est un contenant (container) abstrait fournissant un contexte à des items (mots, noms, termes techniques, etc.)
- Un espace de nompermet de reconnaître sans ambiguïté les items ayant le même nom mais pas le même espace de nom
- Ex:
  - Un répertoire dans un système d'exploitation
  - Un « package » java
  - …

XML

- XML : eXtensible Markup Language
- Langage de balisage extensible conçu pour décrire des données. Les balises XML ne sont pas prédéfinies.
- Métalangage qui nous permet de définir nos propres balises pour nos documents.
- Séparation de la présentation et du contenu
- Namespace: xmlns="http://www.w3.org/1999/xhtml"
- L'objectif d'un espace de nom XML est de :
  - permettre le déploiement de vocabulaires XML (dans lesquels les noms des éléments et attributs sont définis) dans un environnement global
  - de réduire le risque de collisions dans un document donné lorsque des vocabulaires sont combinés

XML - Exemple

```
<liste>
   <employé>
         <nom> Dupont </nom>
         <prénom> Camille </prénom>
         <date_naissance> 23/12/1975 </date_naissance>
   </employé>
   <employé>
         ….
   </employé>
</liste>
```

XML – Exemple avec espace de nommage

```
<mg:livre
      xmlns:bib="http://www.exemple.org/bib"
      xmlns:mg="http://www.polymtl.ca/michelgagnon/"
      xmlns:dc="http://purl.org/dc/elements/1.1">
      <dc:language> es </dc:language>
      <dc:title> Cronicas de Bustos Domecq </dc:title>
      <mg:auteur> Jorge Luis Borges </mg:auteur>
      <mg:auteur> Adolfo Bioy Casares </mg:auteur>
      <dc:publisher> Editorial Losada </dc:publisher>
      <dc:date> 1967 </dc:date>
      <bib:ISBN> 0525475486 </bib:ISBN>
</mg:livre>
```

http://www.professeurs.polymtl.ca/michel.gagnon/Publications/tutorielSWIG04.pdf

XML versus HTML

- XML n'est pas un remplacement d'HTML
- HTML a été conçu pour afficher des données et se concentre surtout sur leur présentation (taille, couleur, etc.)
- XML a été conçu pour décrire des données et se concentre sur la structure de ces données.
- XML a été conçu pour assurer l'interopérabilité
- XML et HTML sont complémentaires
- XSL / XSLT transforme XML en HTML

Technologies XML

- Grande quantité de *parsers* disponibles
- Langages de transformation XSL (XSLT et XSLFO)
  - Langage de transformation de documents XML en d'autres documents XML.
  - Les transformations XSLT ont pour objet de convertir un fichier XML d'un format de document à un autre. Par exemple pour afficher un document XML sur un navigateur web, en le convertissant en XHTML.
  - Une transformation exprimée en XSLT est appelée feuille de style.
- Xlink et Xpointer permettent de référer à une section précise dans un document
- XML utilise un *Document Type Definition (DTD)* ou un *XML Schema* pour définir un modèle des données

XML DTD/ Schema:

- Un document DTD définit les éléments constitutifs d'un document. Il définit la structure syntaxique (un arbre) d'un document type à l'aide d'une grammaire.
- Un document DTD permet donc à toute application de vérifier la conformité d'un document à ce DTD et donc de manipuler et transformer celui-ci lorsqu'il est conforme à ce DTD.

Exemple xml/dtd/xsd : note.xml

```
<?xml version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Exemple xml/dtd/xsd: note.dtd

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

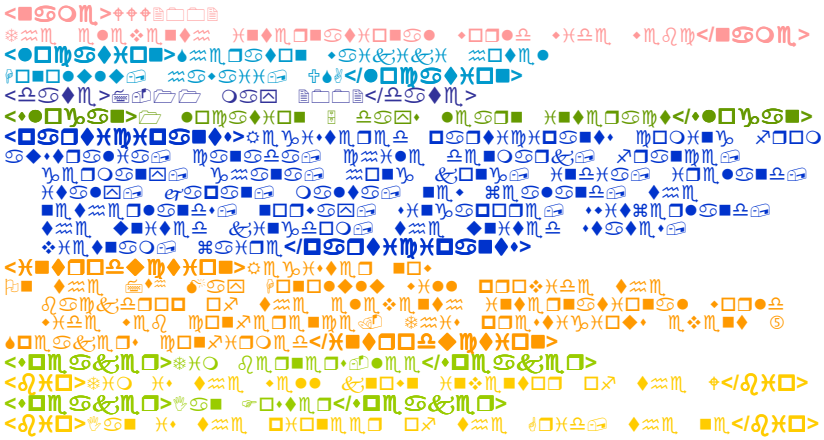
Exemple xml/dtd/xsd: note.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
    <xs:complexType>
      <xs:sequence>
         <xs:element name="to" type="xs:string"/>
         <xs:element name="from" type="xs:string"/>
         <xs:element name="heading" type="xs:string"/>
         <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>

</xs:schema>
```

Est-ce que XML représente LA solution ?

<name>...</name>
<location>...</location>
<date>...</date>
<slogan>...</slogan>
<participants>...</participants>
<introduction>...</introduction>
<speaker>...</speaker>
<bio>...</bio>...

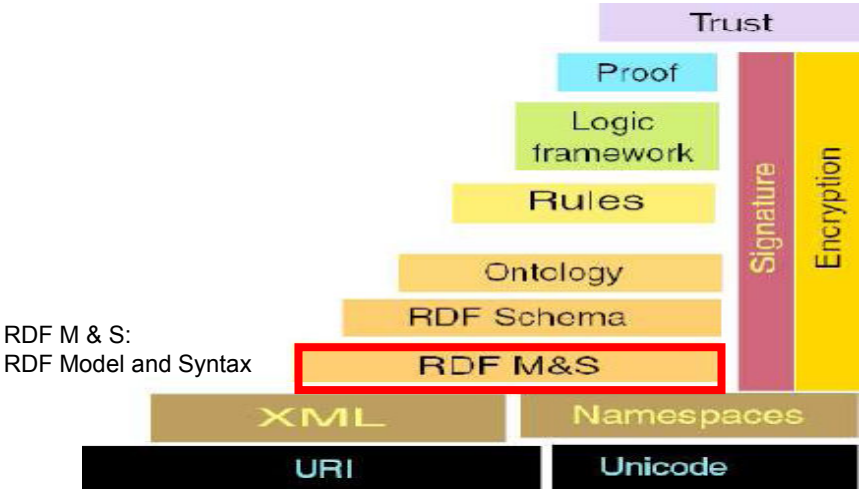
Les ordinateurs voient …



De XML à RDF

- XML ne contient aucune sémantique formelle pour l'ordinateur
- Ce sont les humains qui donnent un sens, une sémantique, aux balises et à leur contenu, pas les machines
- La sémantique permet de définir la signification des balises, donc des informations. Elle contraint les traitements possibles.
- Emergence de RDF, un modèle de données

Les couches du web sémantique



RDF(Resource Description Framework)

- RDF est un langage d'usage général pour représenter l'information sur le Web.
- Simple modèle de données relationnel pour décrire des ressources sur le web sous forme de graphe:
  - les nœuds représentent des ressources
  - les arcs représentent des relations entre ces ressources
- Le graphe est représenté par un ensemble d'énoncés (*statements*)
- Un énoncé est un triplet <S, P, O>, où
  - S est le sujet
  - P est le prédicat (une propriété)
  - O est l'objet (la valeur de la propriété pour le sujet en question)

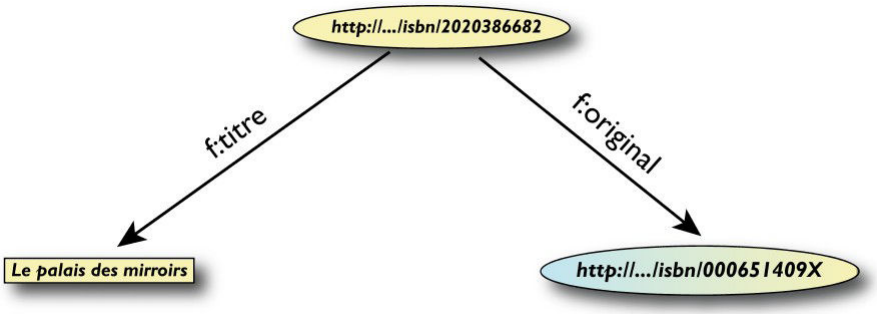
RDF vs XML

- XML fournit une syntaxe pour encoder des données, RDF est un mécanisme qui permet de dire quelque chose au sujet des données.
  - Comme son nom l'indique, ce n'est pas une langage, mais un modèle de représentation des données sur les « objets sur le Web. » (les métadonnées)
- Diverses représentations XML peuvent être utilisées pour représenter la même "chose", pas avec RDF
  - Le modèle de données RDF spécifie que chaque triplet nœud-arc-nœud doit être interprété comme un ensemble de déclarations
  - XML laisse le soin à chacune des spécifications du langage XML de décrire comment la relation parent-enfant et les relations attribut-élément dans l'arbre doivent être interprétées.

Exemple RDF

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
 rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
</rdf:Description>

<rdf:Description
 rdf:about="http://www.recshop.fake/cd/Hide your heart">
  <cd:artist>Bonnie Tyler</cd:artist>
  <cd:country>UK</cd:country>
  <cd:company>CBS Records</cd:company>
  <cd:price>9.90</cd:price>
  <cd:year>1988</cd:year>
</rdf:Description>
.
.
.
</rdf:RDF>
```

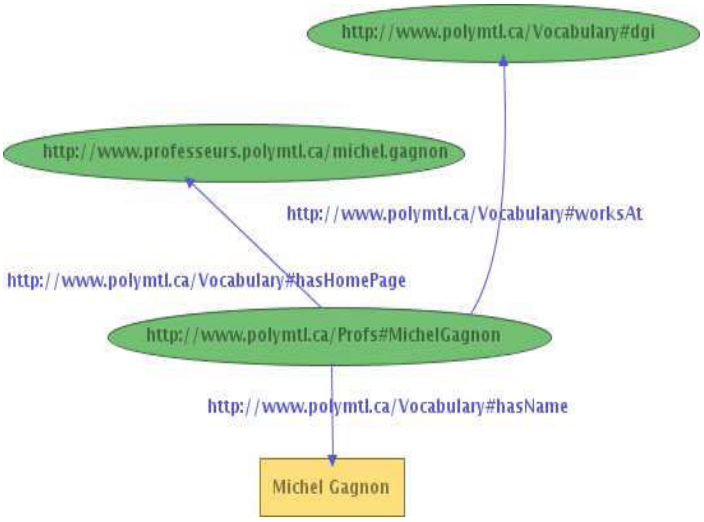
Exemple RDF (en RDF/XML)




```
<rdf:Description rdf:about="http://…/isbn/2020386682">
    <f:titre xml:lang="fr">Le palais des mirroirs</f:titre>
    <f:original rdf:resource="http://…/isbn/000651409X"/>
</rdf:Description>
```

Notation (Syntaxe) RDF

- N-Triples
- Notation 3
- XML



http://www.cours.polymtl.ca/inf6410/Documents/rdf.pdf

40

Syntaxe N-Triple

- Un graphe RDF est représenté par une collection de triplets Sujet Prédicat Objet
- un littéral est représenté directement sans modification
- Si le sujet, le prédicat ou l'objet est une URI, on le représente en mettant entre crochets <> la forme non abrégée de cette URI. Ainsi, on ne peut pas utiliser de préfixe dans la notation N-Triples.

1. <http://www.polymtl.ca/Profs#MichelGagnon>
   <http://www.polymtl.ca/Vocabulary#hasName> « Michel Gagnon »
2. <http://www.polymtl.ca/Profs#MichelGagnon>
   <http://www.polymtl.ca/Vocabulary#worksAt>
   <http://www.polymtl.ca/Vocabulary#dgi> .
3. <http://www.polymtl.ca/Profs#MichelGagnon>
   <http://www.polymtl.ca/Vocabulary#hasHomePage>
   <http://www.professeurs.polymtl.ca/michel.gagnon> .

http://www.cours.polymtl.ca/inf6410/Documents/rdf.pdf

Syntaxe Notation 3 (N3)

- On voit bien que la notation N-Triple est fastidieuse, étant donné l'impossibilité d'abréger les URI
- La syntaxe Notation 3 utilise une notation similaire à la syntaxe N-Triples, sauf qu'elle permet de définir et d'utiliser des préfixes.
- Lorsque qu'une ressource est désignée par une URI dans une forme non abrégée, on la met entre crochets < >. Si on utilise un préfixe, on omet les crochets.
- prof:MichelGagnon local:hasHomePage <http://www.professeurs.polymtl.ca/michel.gagnon> .
- prof:MichelGagnon local:hasName "Michel Gagnon" .
- prof:MichelGagnon local:worksAt local:dgi .

http://www.cours.polymtl.ca/inf6410/Documents/rdf.pdf

Syntaxe RDF/XML

```
<?xml version="1.0"?><rdf:RDF …>
<rdf:Description rdf:about="http://www.polymtl.ca/Profs#MichelGagnon>
  <local:worksAt rdf:resource="http://www.polymtl.ca/Vocabulary#dgi"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.polymtl.ca/Profs#MichelGagnon">
  <local:hasName>Michel Gagnon</local:hasName>
</rdf:Description>
<rdf:Description rdf:about="http://www.polymtl.ca/Profs#MichelGagnon">
  <local:hasHomePage
  rdf:resource="http://www.professeurs.polymtl.ca/michel.gagnon"/>
</rdf:Description></rdf:RDF>
```

http://www.cours.polymtl.ca/inf6410/Documents/rdf.pdf

Modèle logique RDF

- Chaque triplet représente un prédicat binaire en logique

```
(http://udm.ca/doc.html , auteur , urn://~azouaq)
(urn://~azouaq , prénom , "Amal")
(http://udm.ca/doc.html , sujet , "Web sémantique")

auteur(http://udm.ca/doc.html, urn://~azouaq)
prénom(urn://~azouaq, "Amal")
sujet(http://udm.ca/doc.html, "Web sémantique")
```

Modèle logique RDF

- Sémantique formelle: RDF est un sous-ensemble de la logique du premier ordre
  - Avec: prédicats binaires, quantification existentielle(∃),conjonction
  - Sans: disjonction, négation, quantification universelle (∀)
- Tout énoncé RDF est considéré comme vrai et RDF est monotone i.e. ce qui est vrai et ce que l'on peut déduire reste vrai si l'on rajoute de nouveaux énoncés.
- La quantification existentielle (∃) est introduite par les *blank nodes* / nœuds anonymes.

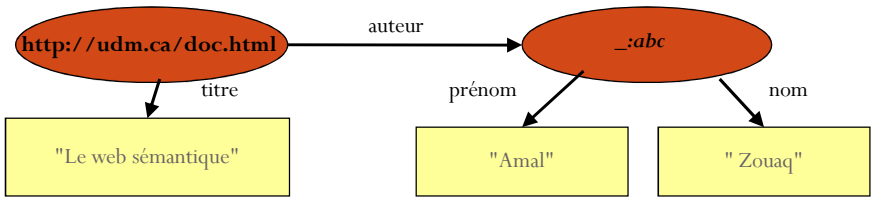
http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Les nœuds anonymes (*blank nodes*)

- Une ressource peut ne pas être identifiée
- sémantique = quantification existentielle
- il existe une ressource telle que… { ∃ r ; …}

```
<rdf:Description rdf:about="http://udm.ca/doc.html ">
  <auteur>
   <rdf:Description>
     <nom>Zouaq</nom>
     <prenom>Amal</prenom>
   </rdf:Description>
  </auteur>
  <titre>Le Web sémantique</titre>
</rdf:Description>
```

∃ x ; auteur(http://udm.ca/doc.html, x)
nom(x, « Zouaq")
prenom(x, « Amal")



Réification

- Réification d'un triplet: rendre un triplet explicite pour pouvoir en parler i.e. l'utiliser comme le sujet ou l'objet d'une propriété.
  - Un triplet est réifié par un *statement*
  - Le *statement* fait du triplet une ressource
  - Cette ressource peut être décrite à son tour

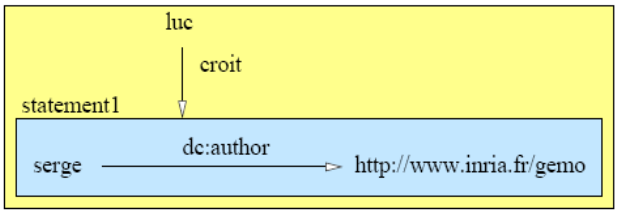
```
<rdf:Statement rdf:nodeID="descAmal">
 <rdf:subject rdf:resource="http://udm.ca/doc.html" />
 <rdf:predicate rdf:resource="&dc;auteur"/>
 <rdf:object rdf:resource="urn://~azouaq"/>
</rdf:Statement>

<rdf:Description rdf:nodeID="descAmal">
  <universite rdf:resource="http://www.umontreal.ca/"/>
</rdf:Description>
```

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Réification (Exemple)

Une déclaration peut aussi être identifiée par une URL : on peut créer déclarations impliquant d'autres déclarations (réification).

(#statement1, rdf:subject, #serge)
(#statement1, rdf:predicate, dc:author)
(#statement1, rdf:object, http://www.inria.fr/gemo)
(#luc, #croit, #statement1)

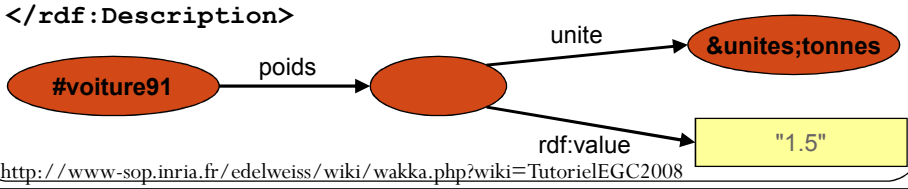


http://www.aristote.asso.fr/Presentations/CEA-EDF-2003/Cours/BerndAmann/Cours_5/cours5.pdf

Valeurs complexes

- Relations n-aires dans le cas d'une valeur littérale *ou* valeur complexe dans une propriété
  - Sélectionner un sujet principal
  - Réifier la relation par une ressource anonyme
  - Déclarer de propriétés pour chaque autre valeur

```
<rdf:Description rdf:about="#voiture91">
  <poids rdf:parseType="Resource">
   <rdf:value rdf:datatype="&xsd;decimal">1.5</rdf:value>
   <unite rdf:resource="&unites;tonnes"/>
  </poids>
</rdf:Description>
```



http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

XML schema datatypes

- XML schema datatypes
  - Les littéraux standards sont des chaînes de caractères
  - Pour typer les valeurs littérales, RDF repose sur les datatypes de XML Schema
- Pour plus d'informations, voir: http://www.w3.org/TR/swbp-xsch-datatypes/

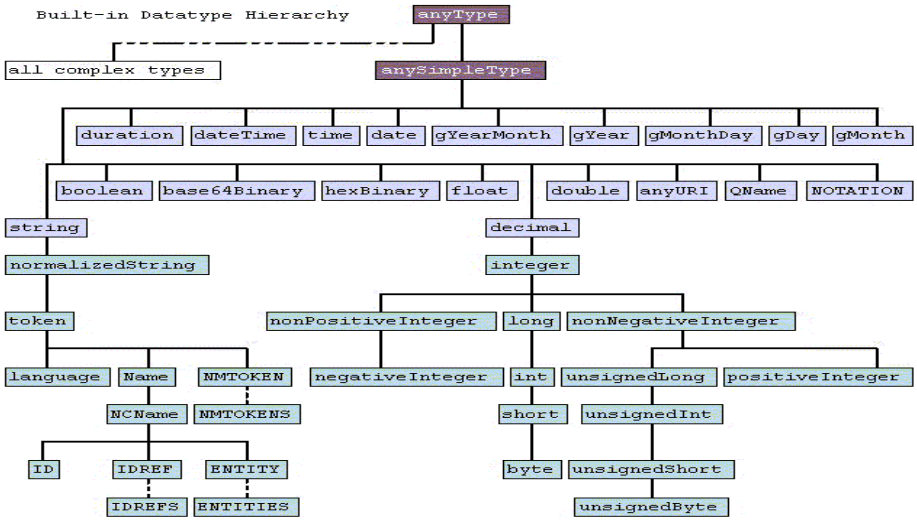
Les datatypes et syntaxe XML

- Syntaxe XML pour les types de données (datatypes) en RDF

```
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
(…)
<rdf:Description rdf:about="#Amal">
 <chargeeDeCours rdf:datatype="&xsd;#boolean">
  true</ chargeeDeCours >
 <naissance rdf:datatype="&xsd;#date">
  1999-07-31</naissance>
(…)
</rdf:Description/>
```

51

XML schema datatypes



http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Groupe simple sans ordre ni sens

- Un groupe de ressources ou littéraux sans ordre

```
<rdf:Description rdf:about="http://www.inria.fr/rrrt/rr-5663.html">
 <auteur>
   <rdf:Bag>
     <rdf:li>Moussa Lo</rdf:li>
     <rdf:li>Fabien Gandon</rdf:li>
  </rdf:Bag>
 </auteur>
</rdf:Description>
```

```
<http://www.inria.fr/rrrt/rr-5663.html> auteur _:a
_:a rdf:_1 "Moussa Lo"
_:a rdf:_2 "Fabien Gandon"
```

53

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Séquence

- Groupe ordonné de ressources ou littéraux

```
<rdf:Description rdf:about="#partition">
  <contient>
   <rdf:Seq>
    <rdf:li rdf:about="#Do"/>   rdf:_1
    <rdf:li rdf:about="#Do"/>   rdf:_2
    <rdf:li rdf:about="#Do"/>   rdf:_3
    <rdf:li rdf:about="#Re"/>   rdf:_4
    <rdf:li rdf:about="#Mi"/>   rdf:_5
   </rdf:Seq>
  </contient>
</rdf:Description>
```

- Accès: `rdf:_1`, `rdf:_2`, `rdf:_3`, `rdf:_4`, etc.

54

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Alternatives

- Groupe de ressources ou littéraux alternatifs
  i.e. une seule valeur est la bonne
  ex: le titre d'un livre en plusieurs langues

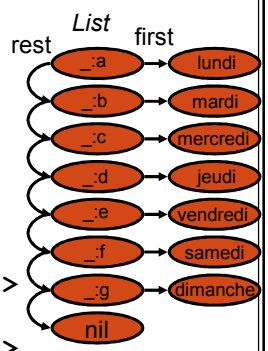
```
<rdf:Description rdf:about="#livre">
  <titre>
   <rdf:Alt>
    <rdf:li xml:lang="fr">l'homme qui prenait sa
femme pour un chapeau</rdf:li>
    <rdf:li xml:lang="en">the man who mistook his
wife for a hat</rdf:li>
   </rdf:Alt>
 </titre>
</rdf:Description>
```

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

55

Collection

- Liste exhaustive et ordonnée de constituants
  (pour fermer une assertion)



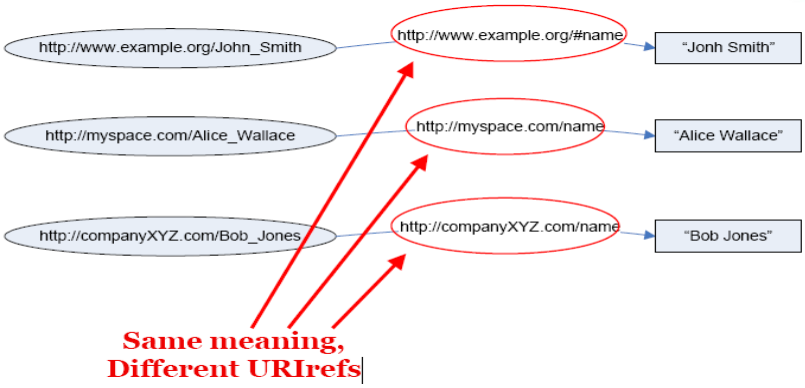

```
<rdf:Description rdf:about="#Semaine">
 <seDiviseEn rdf:parseType="Collection">
  <rdf:Description rdf:about="#Lundi"/>
  <rdf:Description rdf:about="#Mardi"/>
  <rdf:Description rdf:about="#Mercredi"/>
  <rdf:Description rdf:about="#Jeudi"/>
  <rdf:Description rdf:about="#Vendredi"/>
  <rdf:Description rdf:about="#Samedi"/>
  <rdf:Description rdf:about="#Dimanche"/>
 </seDiviseEn>
</rdf:Description>
```

- First / Rest : Le premier et le reste (rdf:List / rdf:nil)

56

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Problème avec RDF



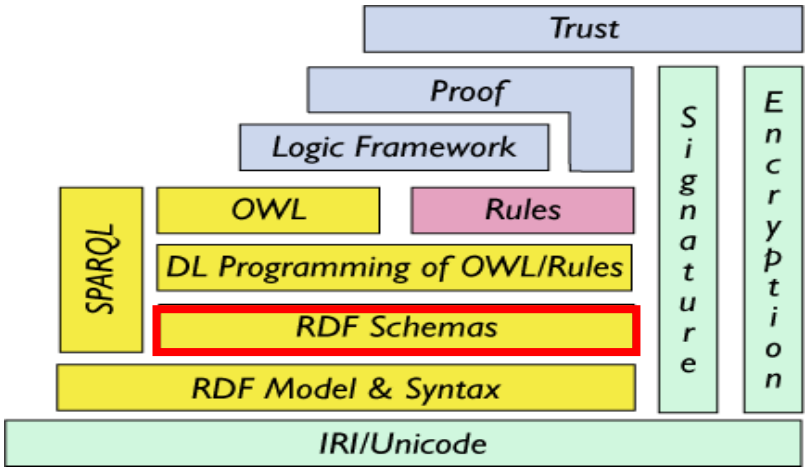
RDF ne décrit pas la sémantique, il permet simplement de déclarer que chaque URI a une sémantique

De RDF à RDF Schéma (RDFS)

- RDF ne permet pas de spécifier le vocabulaire utilisé dans une description RDF, comme par exemple : « author », « music », «creator », etc., c'est-à-dire que RDF ne permet pas de définir la « sémantique » des propriétés.
- RDF Schéma est une extension de RDF avec laquelle il et possible de décrire
  - les concepts utilisés dans des déclarations RDF
  - Un ensemble de contraintes sur les objets et les valeurs du triplet.

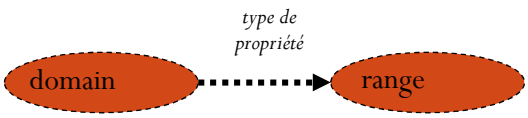
Les couches du web sémantique…

- RDF : modèle de triplets pour annoter des ressources
- RDFS: décrit le vocabulaire utilisé pour ces annotations



Des ontologies légères

- Nommer et définir un vocabulaire conceptuel consensuel et faire des inférences élémentaires
- Nommer les classes de ressources existantes
- Nommer les relations qui existent entre ces classes
- Donner la signature de ces relations:
  - Le domaine (d'où la relation part)
  - La portée (range) (où la relation arrive)

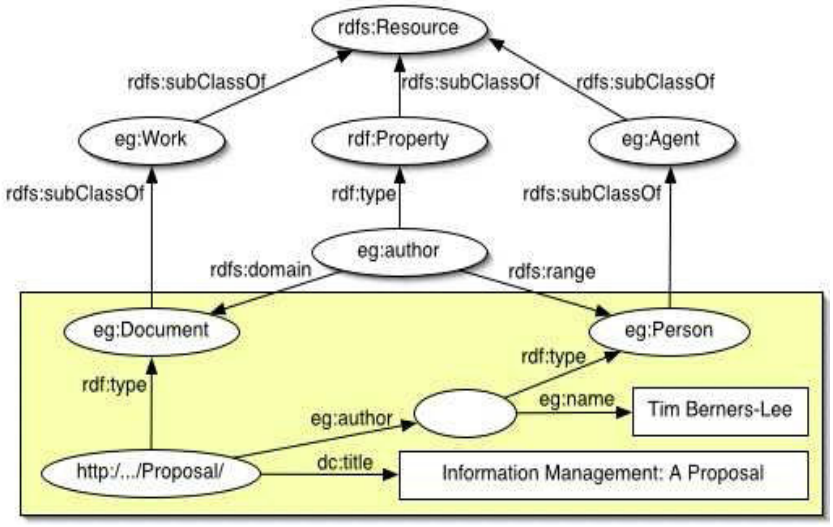


http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Les racines de RDF Schéma

- Tout est ressource.
- Parmi les ressources il y a en particulier…
  - … des classes de ressources qui représentent des types de ressources, des ensembles de ressources;
  - … des propriétés qui représentent des types de relations, des ensembles de relations possibles entre les ressources.
- Parmi les relations il y a en particulier…
  - … la relation de typage / d'instanciation pour dire qu'une ressource/un lien est d'un certain type;
  - … la relation de sous-classes (subsomption) pour dire qu'une classe/propriété est sous classe /propriété d'une autre et que ses instances sont aussi instances de l'autre.

RDF Schéma – exemple 1



RDF Schéma – exemple 2

```
<rdf:RDF xml:base  ="http://inria.fr/2005/humans.rdfs"
 xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"   xmlns
 ="http://www.w3.org/2000/01/rdf-schema#>
 <Class rdf:ID="Man">
   <subClassOf rdf:resource="#Person"/>
   <subClassOf rdf:resource="#Male"/>
   <label xml:lang="en">man</label>
   <comment xml:lang="en">an adult male person</comment>
 </Class>

 <rdf:Property rdf:ID="hasMother">
   <subPropertyOf rdf:resource="#hasParent"/>
   <range rdf:resource="#Female"/>
   <domain rdf:resource="#Human"/>
   <label xml:lang="en">has for mother</label>
   <comment xml:lang="en">to have a female as
 parent.</comment>
 </rdf:Property>
```

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

rdfs:label

- Une ressource peut avoir un ou plusieurs (labels) dans une ou plusieurs langues naturelles

```
<rdf:Property rdf:ID='name'>
 <rdfs:domain rdf:resource='Person'/>
 <rdfs:range rdf:resource='&rdfs;Literal'/>
 <rdfs:label xml:lang='fr'>nom</rdfs:label>
 <rdfs:label xml:lang='fr'>nom de famille</rdfs:label>
 <rdfs:label xml:lang='en'>name</rdfs:label>
</rdf:Property>
```

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

EX d'application de RDF(S) : RSS

- Format standard pour représenter une liste d'informations (nouvelles, bulletin de météo, listes d'offres d'emploi)
- Des programmes sont conçus pour récupérer automatiquement ces informations sur divers sites et les regrouper pour les présenter sur un autre site

EX d'application de RDF(S) : FOAF

- Friend-of-a-friend, une idée originale de Libby Miller et Dan Brickley
- http://www.foaf-project.org
- L'idée consiste à publiciser sur le web une description de soi-même, en quelque sorte une page personnelle lisible par la machine
- Cette description est faite en RDF, en utilisant un vocabulaire prédéfini par les créateurs de FOAF
- FOAF fournit un utilitaire pour créer cette description: FOAF-A-Matic

SPARQL

- Soit une base de données en RDF
- Comment accéder à l'information qui y est contenue?
- Il faut pour cela un langage de requête
- Comme RDF est un modèle de graphe, il est logique que le langage de requête soit basé sur un processus d'appariement de graphes
- W3C propose le langage SPARQL

SPARQL

- Ce que permet SPARQL:
  - Extraire l'information sous forme de URI, de nœuds vides ou de littéraux
  - Extraire des sous-graphes RDF
  - Construire de nouveaux graphes RDF à partir de l'information obtenue

Exemple

SPARQL – Exemple

Base de données RDF:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
_:p1 foaf:name  "Michel Gagnon";
     foaf:mbox  <mailto:michel.gagnon@polymtl.ca> .
_:p2 foaf:name  "Michel Dagenais";
     foaf:mbox  <mailto:michel.dagenais@polymtl.ca> .
```

Requête SPARQL:

```
PREFIX  foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?mbox
WHERE {
          _:a foaf:name  "Michel Gagnon" .
          _:a foaf:mbox  ?mbox .
        }
```

Résultat:

?mbox
<mailto:michel.gagnon@polymtl.ca>

http://www.professeurs.polymtl.ca/michel.gagnon/Publications/tutorielSWIG04.pdf

RDFS

- RDFS permet de décrire
  - Classes et propriétés
  - Sous/super-classes (et propriétés)
  - Portée et domaine (des propriétés)
- Mais RDFS n'est pas suffisant pour décrire des ressources avec suffisamment de détails (voir ci-après)
- RDF(S) a une sémantique non standard
  - Difficile de permettre le raisonnement

Avec RDFS, on peut avoir:

- Hiérarchies de classes:
  - la classe *auto est une sous-classe de la classe moyen de transport*
  - la classe *travailleur autonome est une sous-classe à la fois de travailleur et de contribuable*

http://www.cours.polymtl.ca/inf6410/Documents/tutorielPartie2.pdf

Avec RDFS, on peut avoir:

- Hiérarchies de propriétés:
  - les propriétés *aimer et détester sont des sous-propriétés de la* propriété *éprouver un sentiment*
- les seuls types d'entités qui peuvent *aimer ou détester sont des* instances de la classe *animal*

http://www.cours.polymtl.ca/inf6410/Documents/tutorielPartie2.pdf

Avec RDFS, on ne peut pas avoir:

- Définition de classe par spécification de restrictions sur des propriétés:
  - une *mère est une femme qui a au moins un enfant*
  - un professeur universitaire est une personne *qui enseigne à l'université*
  - un parent heureux est un parent *dont tous les enfants sont heureux*

http://www.cours.polymtl.ca/inf6410/Documents/tutorielPartie2.pdf

Avec RDFS, on ne peut pas avoir:

- Identification de classes disjointes
  - les classes *automobile et autobus sont toutes les deux sous-classes* de *moyen de transport mais sont disjointes (un objet ne* peut être à la fois une automobile et un autobus)

http://www.cours.polymtl.ca/inf6410/Documents/tutorielPartie2.pdf

Avec RDFS, on ne peut pas avoir:

- Définition de classe par combinaison booléenne:
  - une *personne franche est quelqu'un qui est une personne* ***et qui*** **n'est pas un** ***menteur***
  - un *nord-américain est un canadien, un états-unien* ***ou un*** *mexicain*
  - un canadien et quelqu'un *qui est né au canada* ***ou qui a obtenu*** *la citoyenneté*

http://www.cours.polymtl.ca/inf6410/Documents/tutorielPartie2.pdf

Avec RDFS, on ne peut pas avoir:

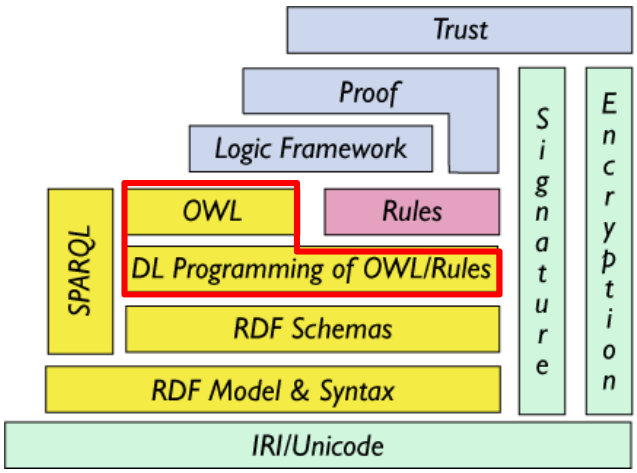
- Caractérisation de certaines propriétés:
  - on ne peut être l'époux de plus d'une personne (propriété définie comme une **fonction)**
  - si X est plus grand queY qui est plus grand que Z, alors X est plus grand que Z (*plus grand est une propriété* ***transitive)***
  - la propriété *parent est l'****inverse de la propriété fils***

http://www.cours.polymtl.ca/inf6410/Documents/tutorielPartie2.pdf

Alors ?!

- On voit donc que pour définir une ontologie vraiment utile, il nous faut plus que le pouvoir expressif de RDF(S)
- De plus: Chacun peut développer son schéma RDFS pour exprimer les liens sémantiques pour son site web.
  - Problème : compatibilité sémantique des connaissances exprimées sur différents sites web
- En fait, il faut pouvoir spécifier un ensemble de restrictions sur les classes qu'on définit
- On aimerait pouvoir faire cela en utilisant un formalisme qui nous permet de faire des inférences
- La réponse à nos besoins: la logique descriptive
- Utilisation des ontologies (extensions de RDFS): OWL

Les couches du web sémantique...

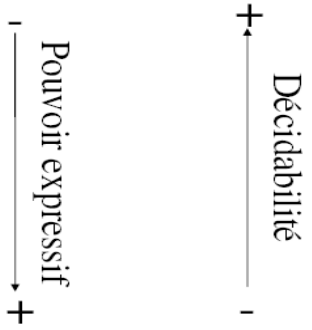


OWL

- Conçu pour des applications qui traitent le contenu, pas uniquement la présentation des informations
- Une extension de RDFS, munie d'une sémantique formelle
- Constitué de trois langages
  - OWL Lite
  - OWL DL
  - OWL Full

OWL: le langage du web sémantique

- OWL Lite
- OWL DL
- OWL Full



http://www.cours.polymtl.ca/inf4215/documentation/tutorielWS.pdf

Rappel

$\mathcal{AL}$	Attributive language. This is the base language which allows: ▪ Atomic negation (negation of concepts that do not appear on the left hand side of axioms) ▪ Concept intersection ▪ Universal restrictions ▪ Limited existential quantification
$\mathcal{FL}^-$	A sub-language of $\mathcal{AL}$, which is obtained by disallowing atomic negation.
$\mathcal{FL}_o$	A sub-language of $\mathcal{FL}^-$, which is obtained by disallowing limited existential quantification.
$\mathcal{C}$	Complex concept negation.
$\mathcal{S}$	An abbreviation for $\mathcal{AL}$ and $\mathcal{C}$ with transitive properties.
$\mathcal{H}$	Role hierarchy (subproperties - rdfs:subPropertyOf).
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.
$\mathcal{O}$	Nominals. (Enumerated classes of object value restrictions - owl:oneOf, owl:hasValue).
$\mathcal{I}$	Inverse properties.
$\mathcal{N}$	Cardinality restrictions (owl:Cardinality, owl:MaxCardinality).
$\mathcal{Q}$	Qualified cardinality restrictions (available in OWL 1.1, cardinality restrictions that have fillers other than owl:thing).
$\mathcal{F}$	Functional properties.
$\mathcal{E}$	Full existential qualification (Existential restrictions that have fillers other than owl:thing).
$\mathcal{U}$	Concept union.
$(\mathcal{D})$	Use of datatype properties, data values or data types.

Parallèle OWL / LD

- OWL-LITE = *SHIF* :
  - *S* : abréviation de AL et C avec des propriétés transitives
  - *H*: hiérarchie de rôles
  - *I*: propriétés inverses
  - *F*: propriétés fonctionnelles ((i.e., 0 ou 1)
- OWL-DL = *SHOIN*
  - *S* : abréviation de AL et C avec des propriétés transitives
  - *H*: hiérarchie de rôles
  - *O:* Individus(classes énumérées avec owl:oneOf et owl:hasValue)
  - *I*: propriétés inverses
  - *N:* Restrictions de cardinalités

OWL Lite

- Classification hiérarchie + contraintes simples
- OWL Lite n'autorise pas:
  - Union
  - Cardinalité autre que 0 ou 1 (par exemple, on ne peut pas définir le concept *personne ayant au moins deux enfants)*
- On ne peut pas définir une classe par énumération (on ne peut pas dire que la classe *pays d'Amérique du nord est définie par* l'ensemble des entités {Canada, USA, Mexique}
- On ne peut pas déclarer des classes disjointes
- Limites sur l'utilisation de rdfs:subClassOf

Constructeurs de classes OWL

A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf($C_1\ C_2 \ldots C_n$)	$C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n$
unionOf($C_1\ C_2 \ldots C_n$)	$C_1 \sqcup C_2 \sqcup \cdots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf($a_1\ a_2 \ldots a_n$)	$\{a_1\} \sqcup \{a_2\} \sqcup \cdots \sqcup \{a_n\}$
restriction($R \ldots$)	
allValuesFrom(C)	$\forall R.C$
someValuesFrom(C)	$\exists R.C$
minCardinality(n)	$\geq n\ R$
maxCardinality(n)	$\leq n\ R$
value(a)	$\exists R.\{a\}$

Les classes

Class(A partial $C_1\, C_2 \ldots C_n$)	$A \sqsubseteq C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n$
Class(A complete $C_1\, C_2 \ldots C_n$)	$A \equiv C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n$
SubClassOf($C\ D$)	$C \sqsubseteq D$
EquivalentClasses($C_1\, C_2 \ldots C_n$)	$C_1 \equiv C_2,\quad C_2 \equiv C_3,\ \ldots,\ C_{n-1} \equiv C_n$
DisjointClasses($C_1\, C_2 \ldots C_n$)	$C_1 \sqsubseteq \neg C_2,\ C_1 \sqsubseteq \neg C_3,\ \ldots,\ C_1 \sqsubseteq \neg C_n$ $C_2 \sqsubseteq \neg C_3,\ \ldots,\ C_2 \sqsubseteq \neg C_n$ $\ldots$ $C_{n-1} \sqsubseteq \neg C_n$

Les propriétés d'objets (Object Properties)

SubPropertyOf($R\ S$)	$R \sqsubseteq S$
EquivalentProperty($R\ S$)	$R \equiv S$
ObjectProperty(R ...)	
super(S)	$R \sqsubseteq S$
inverseOf(S)	$R \equiv S^-$
Transitive	*transitive*(R)
Functional	$\top \sqsubseteq \leq 1\ R$
InverseFunctional	$\top \sqsubseteq \leq 1\ R^-$
Symmetric	$R^- \sqsubseteq R$
range(C)	$\top \sqsubseteq \forall R.C$
domain(C)	$\exists R.\top \sqsubseteq C$

Les contraintes de domaine et de portée (domain et range)

$\text{ObjectProperty}(R\ \text{range}(C))$ $\quad$ $\text{ObjectProperty}(R\ \text{domain}(D))$

$\top \sqsubseteq \forall R.C$ $\quad$ $\exists R.\top \sqsubseteq D$

OWL Lite

RDF Schema Features:

- *Class (Thing, Nothing)*
- *rdfs:subClassOf*
- *rdf:Property*
- *rdfs:subPropertyOf*
- *rdfs:domain*
- *rdfs:range*
- *Individual*

(In)Equality:

- *equivalentClass*
- *equivalentProperty*
- *sameAs*
- *differentFrom*
- *AllDifferent*
- *distinctMembers*

Property Characteristics:

- *ObjectProperty*
- *DatatypeProperty*
- *inverseOf*
- *TransitiveProperty*
- *SymmetricProperty*
- *FunctionalProperty*
- *InverseFunctionalProperty*

Property Restrictions:

- *Restriction*
- *onProperty*
- *allValuesFrom*
- *someValuesFrom*

Restricted Cardinality:

- *minCardinality* (only 0 or 1)
- *maxCardinality* (only 0 or 1)
- *cardinality* (only 0 or 1)

Header Information:

- *Ontology*
- *imports*

Class Intersection:

- *intersectionOf*

Datatypes

- *xsd datatypes*

Versioning:

- *versionInfo*
- *priorVersion*
- *backwardCompatibleWith*
- *incompatibleWith*
- *DeprecatedClass*
- *DeprecatedProperty*

Annotation Properties:

- *rdfs:label*
- *rdfs:comment*
- *rdfs:seeAlso*
- *rdfs:isDefinedBy*
- *AnnotationProperty*
- *OntologyProperty*

OWL DL

- Pouvoir d'expression supérieur, avec complétude (toutes les conclusions sont calculables) et décidabilité
- C'est une logique de description (DL)
- Une classe ne peut pas être instance d'une autre classe
- La plupart des éléments du vocabulaire de RDF(S) ne sont pas valides en OWL-DL
- Toutes les classes doivent être déclarées
- OWL DL pose des contraintes sur l'utilisation de RDF et exige que les classes, propriétés, individus et valeurs de données soient disjointes

OWL FULL

- Expressivité maximale, pas de garantie sur les résultats de calculs (non décidable)
- Liberté totale! Tout ce qui est permis en RDFS l'est aussi en OWL Full
- Tout ce qui est permis en OWL-DL est permis en OWL-Full
- OWL Full permet le mélange de OWL avec RDF Schema et, comme RDF Schema, n'applique pas une stricte séparation des classes, des propriétés, des individus et des valeurs de données

OWL DL et Full

Class Axioms:

- *oneOf, dataRange*
- *disjointWith*
- *equivalentClass* (applied to class expressions)
- *rdfs:subClassOf* (applied to class expressions)

Boolean Combinations of Class Expressions:

- *unionOf*
- *complementOf*
- *intersectionOf*

Arbitrary Cardinality:

- *minCardinality*
- *maxCardinality*
- *cardinality*

Filler Information:

- *hasValue*

91

Equivalences ALCU /OWL-DL

DL	OWL	DL Example
¬ (negation)	owl:complementOf	$\neg Healthy$
⊓ (conjunction)	owl:intersectionOf	$Food \sqcap Sweet$
⊔ (disjunction)	owl:unionOf	$Blue \sqcup \neg Red$
∃ (existential q)	owl:someValuesFrom	$\exists hasChild.Female$
∀ (universal q)	owl:allValuesFrom	$\forall parts.Metal$
⊑ (subsumption)	rdfs:subClassOf	$Employee \sqsubseteq Person$
≡ (equivalence)	owl:equivalenceClass	$Even \equiv \neg Odd$

Ontologies

- Définir de manière déclarative un vocabulaire commun résultat d'un consensus dans un domaine donné
- Chaque élément de vocabulaire possède une interprétation unique partagée par tous les membres du domaine
- Décrire la sémantique des termes et leurs relations
- L'interprétation de chaque terme est unique. Elle est fournie par une sémantique formelle.
- L'ensemble des termes et leurs relations fournissent un cadre interprétatif dépourvu d'ambiguïté pour chaque terme.

Ontologies – caractéristiques

- Restriction de cardinalité
- Définition de classe par spécification de restrictions sur des propriétés
- Identification de classes disjointes
- Définition de classe par combinaison booléenne
- Caractérisation de certaines propriétés (transitivité, fonctions, inverses)

Classe énumérée

- Définition en extension d'une classe i.e. en énumérant tous ses membres

```
<owl:Class rdf:id="CouleurYeux">
    <owl:oneOf rdf:parseType="Collection">
      <owl:Thing rdf:ID="Bleu"/>
      <owl:Thing rdf:ID="Vert"/>
      <owl:Thing rdf:ID="Marron"/>
    </owl:oneOf>
</owl:Class>
```

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Classes définies par union/intersection

- Définition d'une classe par union de classes (utile pour les ranges par exemple)

```
<owl:Class>
  <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Person"/>
     <owl:Class rdf:about="#Group"/>
  </owl:unionOf>
</owl:Class>
```

- Définition complète d'une classe par intersection d'autres classes (équivalence)

```
<owl:Class rdf:ID="Man">
  <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#Male"/>
   <owl:Class rdf:about="#Person"/>
  </owl:intersectionOf>
</owl:Class>
```

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Complément et disjonction & Restriction sur valeur des propriétés

- Définition d'une classe complémentaire

```
<owl:Class rdf:ID="Male">
  <owl:complementOf rdf:resource="#Female"/>
</owl:Class>
```

- Imposer une « disjointness »

```
<owl:Class rdf:ID="Carre">
  <owl:disjointWith rdf:resource="#Rond"/>
</owl:Class>
```

- Contraindre toutes les valeurs:

```
<owl:Class rdf:ID="Herbivore">
 <subClassOf rdf:resource="#Animal"/>
 <subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#eats" />
   <owl:allValuesFrom rdf:resource="#Plant" />
  </owl:Restriction>
 </subClassOf>
</owl:Class>
```

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Restriction sur valeur des propriétés (2)

- Contraindre au moins une valeur:

```
<owl:Class rdf:ID="Sportive">
 <owl:equivalentClass>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hobby" />
    <owl:someValuesFrom rdf:resource="#Sport" />
  </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
```

- Imposer une valeur exacte:

```
<owl:Class rdf:ID="Velo">
 <subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#nbRoues" />
    <owl:hasValue>2</owl:hasValue>
   </owl:Restriction>
 </subClassOf>
</owl:Class>
```

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Restriction sur la cardinalité

- Cardinalité d'une propriété: nombres d'instances différentes d'une propriété *i.e.* nombres de fois où une même ressource est utilisée comme point de départ (domain) d'une propriété avec des valeurs différentes
- Contraintes: nb minimum, nb maximum, nb exact

```
<owl:Class rdf:ID="Person">
 <subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#nom" />
   <owl:maxCardinality>1</owl:maxCardinality>
  </owl:Restriction>
 </subClassOf>
</owl:Class>
```

- La super classe de tout : **`owl:Thing`**
- La classe vide (sans instances) : **`owl:Nothing`**

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Trois types de propriétés

- Les `ObjectProperty` sont des relations entre les ressources uniquement. ex: aPourParent(#thomas,#stéphane)
- Les `DatatypeProperty` ont pour valeur un littéral possiblement typé ex:aPourNom(#thomas,"Thomas")
- Les `AnnotationProperty` sont ignorées dans les inférences, uniquement utilisées pour documenter ou pour des extensions hors des inférences DL

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

Caractéristiques des propriétés

- Propriété symétrique, xRy ⇒ yRx, ex:
  `<owl:SymmetricProperty rdf:ID="hasSpouse" />`
- Propriété transitive, xRy & yRz ⇒ xRz, ex:
  `<owl:TransitiveProperty rdf:ID="hasAncestor" />`
- Propriété fonctionnelle, xRy & xRz ⇒ y=z, ex:
  `<owl:FunctionalProperty rdf:ID="hasMother" />`
- Propriété inversement fonctionnelle,
  xRy & zRy ⇒ x=z, ex:
  `<owl:InverseFunctionalProperty rdf:ID="NumSSociale" />`

Relations d'équivalence & Gestion de l'ontologie

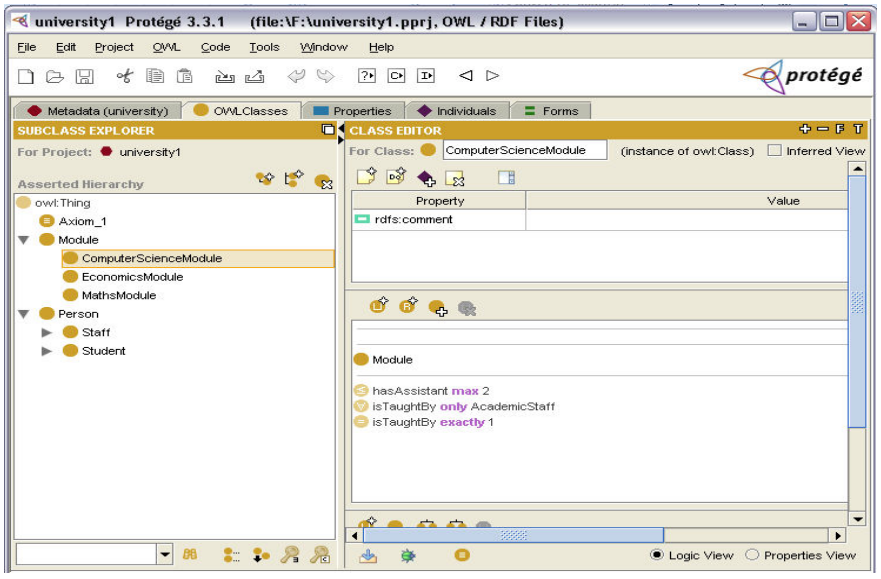
- Classes équivalentes: owl:equivalentClass
- Propriétés équivalentes: owl:equivalentProperty
- Instances identiques ou différentes: owl:sameAs, owl:differentFrom
- Deux propriétés inverses, $xR_1y \Leftrightarrow yR_2x$, ex:
  <rdf:Property rdf:ID="hasChild">
  <owl:inverseOf rdf:resource="#hasParent"/>
  </rdf:Property>
- Utilité dans la mise en correspondance d'ontologies:
  <owl:Class rdf:about="&o1;Person">
  <owl:equivalentClass rdf:resource="&o2;Hito"/>
  </owl:Class>
- Description de l'ontologie:
  owl:Ontology, owl:imports, owl:versionInfo, owl:priorVersion, owl:backwardCompatibleWith, owl:incompatibleWith
- Versions des classes et des propriétés: owl:DeprecatedClass, owl:DeprecatedProperty

Structure d'une ontologie

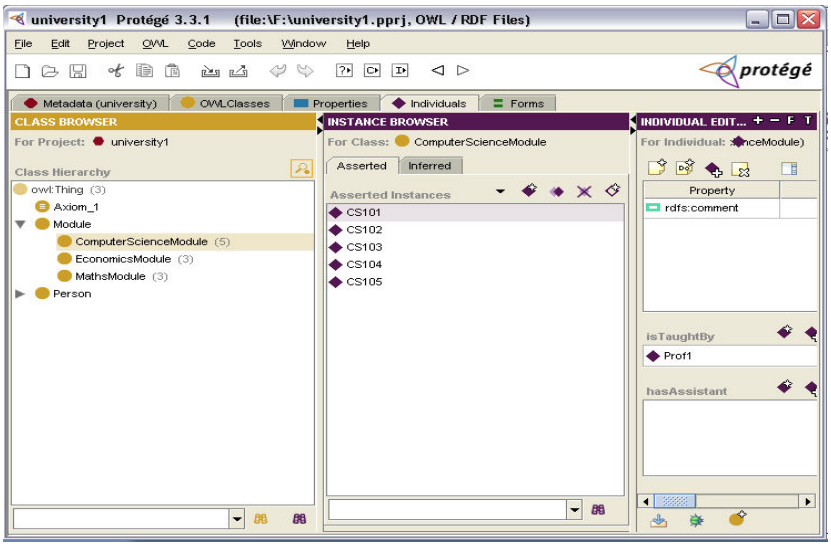
Les ontologies ont typiquement deux composants distincts:

- Noms pour les concepts importants du domaine
  - Elephant is a concept whose members are a kind of animal
  - Herbivore is a concept whose members are exactly those animals who eat only plants or parts of plants
  - Adult_Elephant is a concept whose members are exactly those elephants whose age is greater than 20 years
- De la connaissance de base et des contraintes sur les concepts
  - Adult_Elephants weigh at least 2,000 kg
  - All Elephants are either African_Elephants or Indian_Elephants
  - No individual can be both a Herbivore and a Carnivore

Ontologie exemple (TBOX)(Protégé)



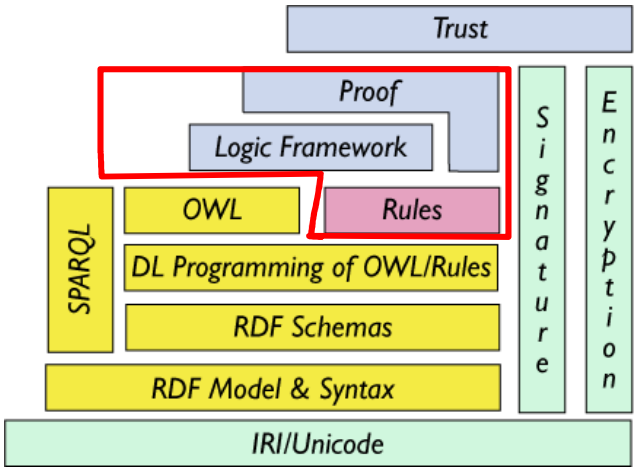
Ontologie exemple (ABOX)(Protégé)



Web sémantique + Ontologies

- En créant une ressource web, on spécifie l'ontologie utilisée
- Une ontologie standard est déposée à un endroit du web
- Toutes les ressources web utilisant la même ontologie parlent maintenant le même langage
- Possibilité d'échanges automatiques
- Communications machine à machine
  - Par exemple : B2B : Business to Business
- Une ontologie du domaine est limitée à un domaine d'application
- Problème de la définition d'ontologies standards
  - Emergence d'ontologies approuvées avec le temps

Les couches du web sémantique…



Logique et preuves

- Il faut pouvoir faire des inférences.
- Il faut aussi pouvoir les expliquer
- Nous avons déjà vu que :
  - La logique est un langage permettant d'exprimer des « règles » de raisonnement
  - Ces règles permettent de déduire de nouveaux faits à partir des faits existants
  - Une preuve est en quelque sorte une suite d'applications de règles qui permettent de déduire un nouveau fait

Pourquoi des règles dans le WS ?

- Il existe des conditions que les ontologies(i.e., OWL) ne peuvent pas exprimer
  - Règles de Horn: (P1 ∧ P2 ∧ …) → C
- Formalisme différent: peut-être plus facile pour certaines personnes

Exemple de règles

- SI instance(X, Professor) ET possede_diplome(X, PhD) ALORS peut_etre_membre(X, JuryDoctorat)
- SI peut_etre_membre(X, JuryDoctorat) ET disponible(X) ALORS membre_potentiel(X, JuryDoctorat)

http://www.cours.polymtl.ca/inf4215/documentation/tutorielWS.pdf

Autre exemple

- "Si deux personnes ont le même nom et le même email , ou le même nom et la même page web alors ces personnes sont identiques"

```
If  {  ?x rdf:type foaf:Person.
       ?y rdf:type foaf:Person.
       ?x foaf:name ?n.
       ?x foaf:homepage ?h.
       ?y foaf:name ?n.
       ?y foaf:homepage ?h.  }
then  {  ?x = ?y  }

If  {  ?x rdf:type foaf:Person.
       ?y rdf:type foaf:Person.
       ?x foaf:name ?n.
       ?x foaf:mailbox ?h.
       ?y foaf:name ?n.
       ?y foaf:mailbox ?m.  }
then  {  ?x = ?y  }
```

SWRL

- SWRL: Semantic Web Rule Language : Combinaison de OWL et Rule Markup Language
- Pas encore un standard
- SWRL permet aux utilisateurs de rédiger des règles pour raisonner sur les individus OWL et d'inférer de nouvelles connaissances sur ces individus
- Inférence monotone

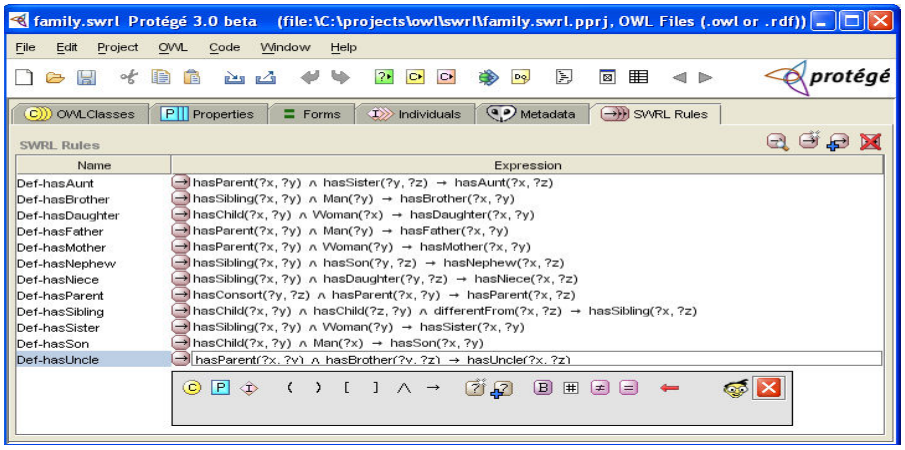
Exemple de règles

- Exemple de regle: hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#example1"/>
 <ruleml:_body>
   <swrlx:individualPropertyAtom  swrlx:property="hasParent">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x2</ruleml:var>
   </swrlx:individualPropertyAtom>
   <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
     <ruleml:var>x2</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
   <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

SWRLTab (Protégé)

- SWRLTab est une extension de Protégé qui permet l'édition et l'exécution de règles SWRL.



Raisonner avec des règles SWRL

- Incorporation d'un moteur de règles tel que Jess pour raisonner sur des règles SWRL
- Détails:
  - http://smi.stanford.edu/smi-web/reports/SMI-2005-1080.pdf

Niveau de confiance

- Si on utilise des agents pour prendre des décisions à notre place, il faudrait qu'on puisse avoir « confiance » dans les résultats fournis
- Pour ce faire, il faut que l'agent utilisé puisse:
  - expliquer clairement comment il arrive à ses conclusion (preuve)
  - garantir la fiabilité et l'origine des informations utilisées (signature digitale)

Niveau de confiance

- Si une personne affirme que x est bleu, et qu'une autre dit que x n'est pas bleu, est-ce que l'ensemble du Web sémantique s'écroule?

  La réponse est évidemment non, parce que :
  - a) les applications sur le Web sémantique pour le moment dépendent généralement d'un contexte
  - b) parce que les applications à l'avenir contiendront généralement des mécanismes de vérification de preuve, et des signatures numériques.

Défis à venir

Augmenter l'expressivité

- OWL n'est pas suffisamment expressif pour certaines applications
  - Constructeurs essentiellement pour les classes (prédicats unaires)
  - Pas de types de données complexes ou de prédicats intégrés (*built in predicates* : arithmétique par exemple)
  - Pas de variables
  - Pas de prédicats d'arité supérieure
- Extensions (de OWL) considérées:
  - extensions (décidables) à la LD
  - Extensions basées sur les règles
  - Logique du premier ordre (ex: SWRL-FOL)
  - …

Extensions de OWL par des règles

- Extension du premier ordre déjà développées (SWRL)
  - Clauses de Horn où les prédicats sont des classes et des propriétés OWL
  - Le langage résultant est non décidable
  - Raisonnement via des démonstrateurs de preuves de la logique de premier ordre (Hoolet)
- …

Améliorer la mise à échelle

- Les ontologies peuvent être de taille très importante
- Bonne évidence empirique sur la mise à échelle du raisonnement avec les systèmes LD
- Le raisonnement utilisant les individus peut s'avérer problématique
  - Bcp d'individus: les techniques standards existantes risquent de ne pas suffire

Autres tâches de raisonnement

- Les requêtes
  - La recherche et l'instanciation ne sont pas des opérations suffisantes
  - Des langages de requêtes similaires à ceux des bases de données vont être requis
- Explication
  - Pour aider à la modélisation d'ontologies
  - Pour présenter des justifications et des preuves (de résultats de requêtes)

Autres défis

- Indexation, appariement et évaluation des ontologies
- Création, documentation et maintenance des ontologies (outils de traitement de la langue naturelle entre autres)
- Automatisation du processus d'annotation
- Développement d'agents intelligents

Le contenu de ces diapositives est tiré de:

- http://www.ifi.auf.org/personnel/Alain.Boucher/cours/intelligence_artificielle/07-Representation_de_connaissances.pdf
- http://www.cours.polymtl.ca/inf4215/documentation/tutorielWS.pdf
- http://www.cours.polymtl.ca/inf6410/Documents/tutorielPartie2.pdf
- http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008
- http://www.w3.org/People/Ivan/CorePresentations/RDFTutorial/Slides.pdf
- http://www.w3schools.com/rdf/rdf_example.asp
- http://www.aristote.asso.fr/Presentations/CEA-EDF-2003/Cours/BerndAmann/Cours_5/cours5.pdf
- http://nfig.hd.free.fr/util/Referential/Ontology/Tutorial%20OWL%20and%20Description%20Logic/Sew06-10.pdf