Web sémantique

Version 2012

Pierre-Antoine Champin

18 January 2016

Table des matières

CHAPITRE 1

Logiques de descriptions

author Pierre-Antoine Champin

1.1 Historique

1.1.1 Logique

— Logique des propositions (Boole)
— Logique des prédicats (Frege, ...)
— Calculabilité, complexité (Turing-Church)
— Incomplétude (Gödel)

1.1.2 Règles

— Clauses de Horn
— Systèmes experts (PROLOG)

```
cretois(minos) .
menteur(X) :- cretois(X) .

?- menteur(minos)
yes
?- menteur(M)
M=minos
```

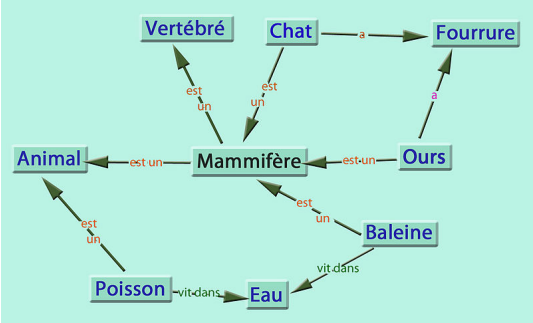
— Difficulté à maintenir de grosses bases de règles

1.1.3 Logiques « intuitives »

— Réseaux sémantiques
  — représentation graphique

Logiques « intuitives » (suite)

— *Frames* (Minsky)
  — approche « objet »
  — stéréotypes



— valeurs par défaut
— logique non monotone
(tous les oiseaux volent, les autruches ne volent pas)

1.1.4 Remise en ordre

— graphes conceptuels (Sowa)
  — représentation graphique formalisée
— logiques de descriptions
  — représentation logique simplifiée

1.1.5 Logiques de description

— Pas un langage, mais une *famille* de langage
— Formalisme de plus haut niveau que la LPO
— Compromis maîtrisé entre décidabilité et complexité
— Autres appellations :
  — logiques descriptives
  — logiques terminologiques

1.2 Syntaxe et sémantique

1.2.1 Éléments de base

L'univers du discours est constitué d'**individus**, appartenant à des **concepts** (ou classes), et reliés entre eux par des **rôles** (ou propriété).

LD	LPO
Individu	Terme
Concept	Prédicat 1-aire
Rôle	Prédicat 2-aire

1.2.2 Exemples

Concepts	Rôles	Individus
Homme	connait	john
Voiture	(a pour) père	jane
Rouge	mère	ab-123-cd
Menteur	enfant	
Ferrari	conduit	

1.2.3 Axiomes

Syntaxe	Appellation	Sémantique
$C \sqsubseteq D$	subsomption de concepts	$\forall$ x, C(x) $\rightarrow$ D(x)
$r \sqsubseteq s$	subsomption de rôles	$\forall$ x, y, r(x, y) $\rightarrow$ s(x, y)

NB : l'*équivalence* peut s'exprimer par deux subsomptions symétriques : $C \sqsubseteq D$ et $D \sqsubseteq C$

Axiomes : exemples

— Crétois $\sqsubseteq$ Menteur
— ami $\sqsubseteq$ connait

1.2.4 Concepts complexes : constructeurs ensemblistes

Syntaxe	Appellation	Sémantique
$\top$	concept universel (*top*)	Δ
$\bot$	concept absurde (*bottom*)	$\varnothing$
$\neg C$	complément	{ x | $\neg$C(x) }
$C \sqcup D$	union	{ x | C(x) } $\cup$ { x | D(x) }
$C \sqcap D$	intersection	{ x | C(x) } $\cap$ { x | D(x) }
{a}	extension	{a}

Constructeurs ensemblistes : exemples

— ¬Menteur
— Homme $\sqcup$ Voiture
— Crétois $\sqcap$ Menteur
— Voiture $\sqcap$ (Rouge $\sqcup$ ¬Ferrari)
— {john, paul, george, ringo}

1.2.5 Concepts complexes : restrictions

Syntaxe	Appellation	Sémantique
$\exists$ r C	qualificateur existentiel	{ x | $\exists$ y, r(x, y) $\wedge$ C(y) }
$\forall$ r C	qualificateur universel	{ x | $\forall$ y, ¬r(x, y) $\vee$ C(y) }
= n r C	quantificateur	{ x | #{y | r(x, y) $\wedge$ C(y)} = n }
$\leq$ n r C	quantificateur (max)	{ x | #{y | r(x, y) $\wedge$ C(y)} $\leq$ n }
$\geq$ n r C	quantificateur (min)	{ x | #{y | r(x, y) $\wedge$ C(y)} $\geq$ n }

NB : on omet généralement C lorsqu'il s'agit de $\top$; *e.g.*

$\exists$ r , $\forall$ r , = 1 r ...

Restrictions : exemples

- $\exists$ enfant
- $\forall$ conduit Ferrari
- ($\exists$ conduit) $\sqcap$ ($\forall$ conduit Ferrari)
- = 2 conduit Ferrari
- $\geq$ 2 connait (Crétois $\sqcap$ Menteur)

1.2.6 Rôles complexes

Syntaxe	Appellation	Sémantique
r^-	rôle inverse	{ (x, y) | r(y, x) }
r$\circ$s	rôle composé	{ (x, y) | $\exists$ z, r(x, z) $\wedge$ s(z, y) }
$\neg$r	complément	{ (x, y) | $\neg$r(x, y) }

Exemples

- conduit$^-$
- connait $\circ$ conduit
- connait $\circ$ connait

1.2.7 Axiomes complexes : exemples

- $\exists$ conduit $\sqsubseteq$ Adulte
- $\exists$ conduit$^-$ $\sqsubseteq$ Voiture
- Personne $\sqsubseteq$ Adulte $\sqcup$ Enfant
- Personne $\sqsubseteq$ (= 1 père Homme) $\sqcap$ (= 1 mère Femme)
- Personne $\sqsubseteq$ $\neg$Voiture

1.2.8 Décidabilité et complexité

Chaque LD impose des contraintes sur :
- les axiomes autorisés,
- les constructeurs autorisés,
- la manière de les combiner,

afin de garantir que les mécanismes de raisonnement
- seront décidables,
- auront une complexité maximale.

$\rightarrow$ compromis entre expressivité et complexité.

1.3 Raisonnement

1.3.1 Rappels

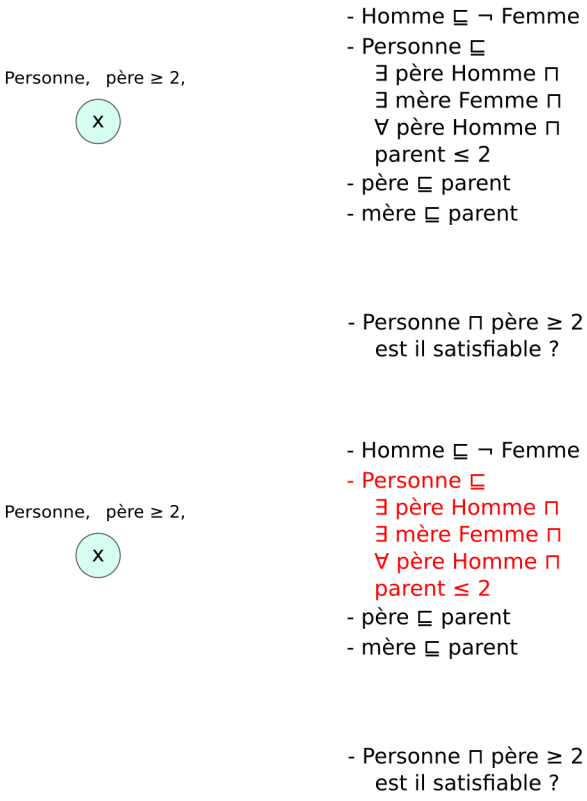
- Interprétation de F
  - domaine du discours

  - fonction d'interprétation
- Modèle
  - axiomes → contraintes
- Implication
  - F est satisfiable si elle a au moins un modèle
  - F $\models$ G ssi tous les modèles de F sont des modèles de G
  - → F $\models$ G ssi F $\wedge$ ¬G est non satisfiable

1.3.2 Méthode des tableaux

- Un tableau est un arbre, représentant une famille de modèles
- Application systématique de règles pour explorer tous les modèles possibles (*non déterministe*)
- Preuve par **réfutation** : concept non satisfiable

1.3.3 Exemple



∃ père Homme, ∀ père Homme,
∃ mère Femme, parent ≤ 2
Personne, père ≥ 2,

- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

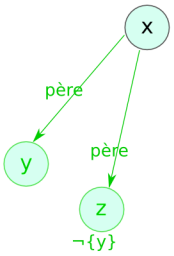
- Personne ⊓ père ≥ 2
  est il satisfiable ?

∃ père Homme, ∀ père Homme,
∃ mère Femme, parent ≤ 2
Personne, père ≥ 2,

- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?

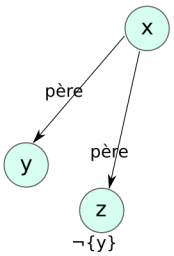
∃ père Homme, ∀ père Homme,
∃ mère Femme, parent ≤ 2
Personne, père ≥ 2,



- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?

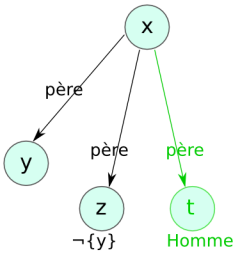
∃ père Homme, ∀ père Homme,
∃ mère Femme, parent ≤ 2
Personne, père ≥ 2,



- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?

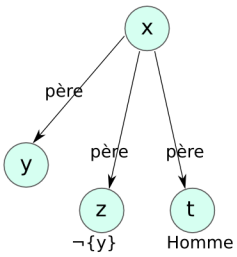
∃ père Homme, ∀ père Homme,
∃ mère Femme, parent ≤ 2
Personne, père ≥ 2,



- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?

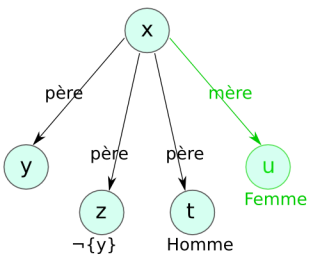
∃ père Homme, ∀ père Homme,
∃ mère Femme, parent ≤ 2
Personne, père ≥ 2,



- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?

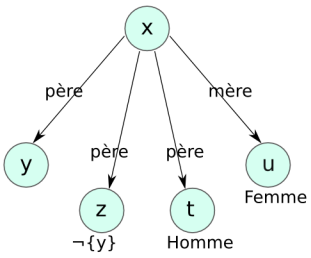
∃ père Homme, ∀ père Homme,
∃ mère Femme, parent ≤ 2
Personne, père ≥ 2,



- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

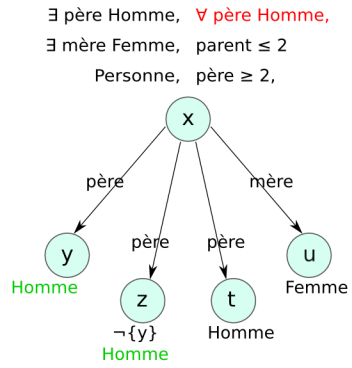
- Personne ⊓ père ≥ 2
  est il satisfiable ?

∃ père Homme, ∀ père Homme,
∃ mère Femme, parent ≤ 2
Personne, père ≥ 2,



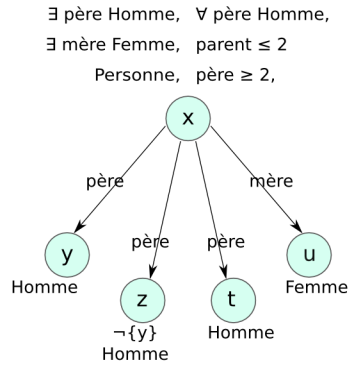
- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?



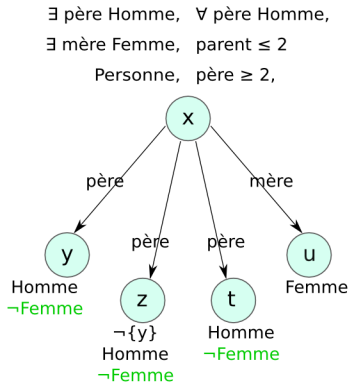
- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?



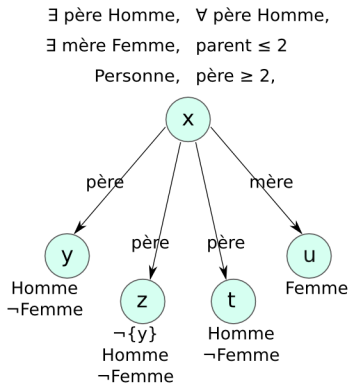
- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?



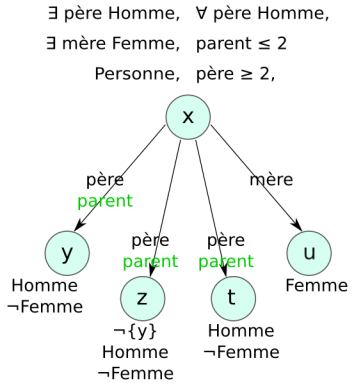
- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?



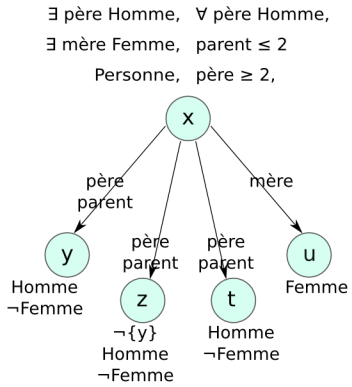
- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?



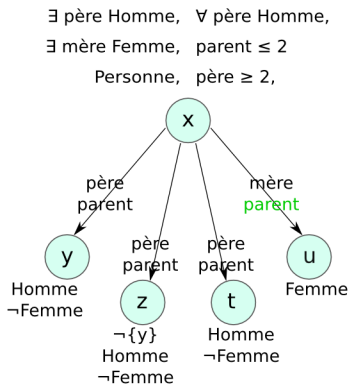
- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?



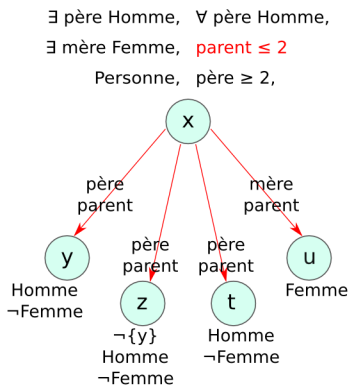
- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?



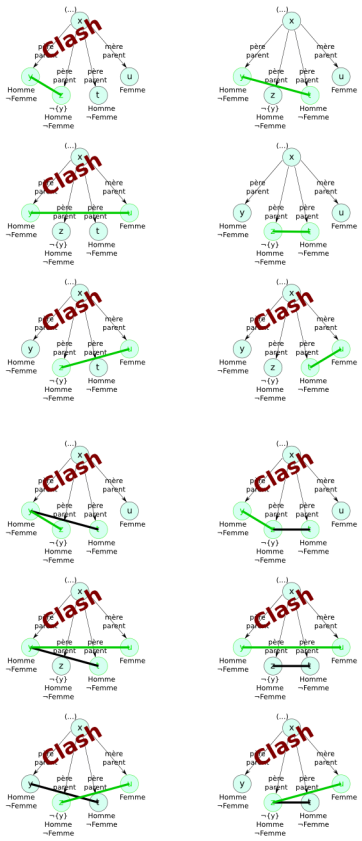
- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?



- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?



- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  est il satisfiable ?

- Homme ⊑ ¬ Femme
- Personne ⊑
  ∃ père Homme ⊓
  ∃ mère Femme ⊓
  ∀ père Homme ⊓
  parent ≤ 2
- père ⊑ parent
- mère ⊑ parent

- Personne ⊓ père ≥ 2
  n'est pas satisfiable

1.3.4 Enjeux

— Décidable : dépend du type de LD choisie
— Correct/complet : ensemble de règles de transformation
— Complexe : optimiser l'ordre d'application des règles
— Problème des modèles infinis : exemple
  — Entier ⊑ (= 1 suivant Entier) ⊓ (≤ 1 suivant$^-$)
  — {zero} ⊑ Entier ⊓ (= 0 suivant$^-$)

1.3.5 Implémentations

Hermit http://hermit-reasoner.com/

Pellet http://clarkparsia.com/pellet

Racer http://www.racer-systems.com/

FaCT http://www.cs.man.ac.uk/~horrocks/FaCT/

1.3.6 Méta-modélisation

— En théorie : séparation stricte entre les individus, les concepts et les rôles
— En pratique : pas d'ambiguïté syntaxique sur la nature d'un terme
— *Punning* (calembour) : autorisation d'utiliser le même terme pour des éléments de nature différente
  — aucun lien sémantique entre eux
  — mais intérêt *pragmatique*

— /!\ pas très bien supporté par Protégé :-(

1.4 Annexe : Protégé

1.4.1 Installation

Téléchargement de Protégé (version ≥ 4)

http://protege.stanford.edu/

Prise en main

http://protegewiki.stanford.edu/wiki/Protege4GettingStarted

1.4.2 Syntaxe de Protégé

Protégé utilise une syntaxe alternative
- inspirée de la syntaxe Manchester
- n'utilisant que l'alphabet latin
- donnes des phrases en pseudo-anglais $\rightarrow$ lisibilité

Concepts complexes : constructeurs ensemblistes

Protégé	LD
Thing	$\top$
Nothing	$\bot$
C and D	C $\sqcap$ D
C or D	C $\sqcup$ D
not C	$\neg$C
{a}	{a}

Concepts complexes : restrictions

Protégé	LD
r some C	$\exists$ r C
r only C	$\forall$ r C
r exactly n C	= n r C
r max n C	$\leq$ n r C
r min n C	$\geq$ n r C

Rôles complexes

Protégé	LD
inverse(r)	r^-
r o s	r$\circ$s

1.4.3 Axiomes dans Protégé-OWL

Protégé offre des axiomes « de haut niveau » qui visent à
- améliorer la lisibilité de la base de connaissance
- optimiser le raisonnement
- contraindre l'utilisation de certains constructeurs à certaines formes d'axiome selon les profils (*e.g.* Property Chain)

NB : ces axioms viennent en fait du langage OWL.

Sur les concepts

Protégé	LD
Equivalent class(C,D)	$C \sqsubseteq D$ et $D \sqsubseteq C$
Super class(C,D)	$C \sqsubseteq D$
Member(C,a)	C(a)
Disjoint class(C,D)	$C \sqsubseteq \neg D$

Sur les rôles

Protégé	LD
Functional(r)	$\top \sqsubseteq (\leq 1\ r)$
Inverse functional(r)	$\top \sqsubseteq (\leq 1\ r^-)$
Transitive(r)	$r \circ r \sqsubseteq r$
Symmetric(r)	$r \sqsubseteq r^-$
Asymmetric(r)	$r \sqsubseteq \neg(r^-)$
Reflexive(r)	$\top \sqsubseteq (\exists\ r\ self)$
Irreflexive(r)	$\top \sqsubseteq \neg(\exists\ r\ self)$

Sur les rôles (suite)

Protégé	LD
Domain(r,C)	$\exists\ r \sqsubseteq C$
Range(r,C)	$\exists\ r^- \sqsubseteq C$
Equivalent property(r,p)	$r \sqsubseteq p$ et $p \sqsubseteq r$
Super property(r,p)	$r \sqsubseteq p$
Inverse property(r,p)	$r \sqsubseteq p^-$ et $p^- \sqsubseteq r$
Disjoint property(r,p)	$r \sqsubseteq \neg p$
Property chain(r,p,q...)	$p \circ q \circ ... \sqsubseteq r$

1.5 TP

http://champin.net/2014/iade1-tp-ld/

CHAPITRE 2

RDF : Resource Description Framework

author Pierre-Antoine Champin

2.1 Motivation et historique

2.1.1 Le Web vu par Tim Berners-Lee (1989)

« Vague, but exciting »

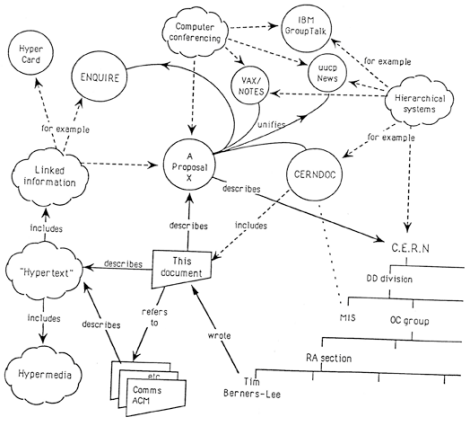


Fig. 2.1 – source : http://info.cern.ch/images/proposal.gif

2.1.2 Web de ressources

Le web est constitué de **ressources** :

- chaque ressource est identifiée par un URI (*Uniform Resource Identifier*), *e.g.* :
  - http://meteo.example.com/lyon
  - http://meteo.example.com/ici
  - http://commerce.example.com/commande/192837

  NB : ressource $\neq$ fichier !
- chaque ressource a un *état* qui peut varier dans le temps
  - cet état n'est jamais manipulé directement, mais toujours à travers des **représentations**

Ressources et représentations

— À chaque état d'une ressource peuvent correspondre une *ou plusieurs* représentation(s) (négociation de contenu, contexte)
— Toute interaction avec une ressource se fait *via* des représentation :
consultation, modification, création

représentation :	utilisable par :
texte	humains, moteurs de recherche
médias (image, son...)	*surtout* humains
données structurées	machines

2.1.3 De HTML à XML

XML (eXtensible Markup Language) a été recommandé par le W3C en 1998. L'objectif était de pallier la sémantique « faible » de HTML.

```
<!-- HTML -->
<a href="http://champin.net/">
  Pierre-Antoine <strong>Champin</strong>
  (<em>Maître de conférences</em>)</a>
```

```
<!-- XML -->
<Person homepage="http://champin.net/">
  <givenName>Pierre-Antoine</givenName>
  <surname>Champin</surname>
  <job>Maître de conférences</job></Person>
```

2.1.4 XML et la sémantique

On a dit tout et son contraire l'apport sémantique de XML :
— XML a *plus* de sémantique que HTML,
— XML a *moins* de sémantique que HTML,
Les deux ont leur part de vérité.

XML a *plus* de sémantique que HTML...

... dans le sens ou il est extensible : on eut donc exprimer des choses que HTML ne permet pas d'exprimer (e.g.‘‘<firstname>‘‘).
— Importance des *espaces de noms*, qui évitent les collisions de noms et fournissent ainsi une sémantique « structuraliste » (i.e. par différentiation).

```
<Person xmlns="http://xmlns.com/foaf/0.1/"
        xmlns:pro="http://example.com/"
     homepage="http://champin.net/">
  <givenName>Pierre-Antoine</givenName>
  <surname>Champin</surname>
  <pro:job>Maître de conférence</pro:job></Person>
```

XML a *moins* de sémantique que HTML...

... dans la mesure ou :
- un navigateur standard ne saura pas quoi faire de la balise `<firstname>` ou de la balise `<ονομα>`,
  - tout au plus il saura les afficher s'il possède une feuille de style,
- tandis qu'il connaît la sémantique de la balise `<em>` : elle dénote un texte à mettre en évidence *selon les moyens dont il dispose*, par exemple :
  - en le mettant en italique (standard)
  - en le mettant en gras (police déjà en italique)
  - en le mettant en couleurs (police sans italique, terminal)
  - en marquant une pause (synthèse vocale)

XML : apports et limitations

Le surplus de sémantique promis par XML n'est donc pas « magique » : il suppose
- de créer de nouveaux langages basés sur XML (DTD, schémas),
- d'écrire les logiciels qui *interpréteront* ces nouveaux langages,
  → chaque langage reste relativement idiosyncratique.

XML : apports et limitations (suite)

L'apport est donc essentiellement technique : la base commune de XML permet de *factoriser* les efforts de développement et d'apprentissage :
- analyseurs syntaxiques (*parsers*),
- langages de schéma (DTD, XML-Schema, Relax-NG...),
- langages de requêtes (XPath, XQuery),
- langages de transformation (XSL-T),
- méthode de signature cryptographique (xmldsig),
- methode de compression (EXI)...

2.1.5 De XML à RDF

- Le modèle sous-jacent de la syntaxe XML est un arbre (*XML Infoset*), ce qui n'est pas adapté à la structure décentralisée du Web.
- L'objectif du *Resource Description Framework* (RDF), recommandé par le W3C en 1999, vise à munir le Web d'un modèle de données plus adapaté, ayant une structure de *graphe*.
- L'objectif est de construire le *Semantic Web* : un web dans lequel les machines ont (enfin) accès à la sémantique des données.
- Recommandation un peu hâtive, présentant quelques défauts importants (notamment l'absence de sémantique formelle).
  → faible adoption de RDF

2.1.6 De RDF à RDF

- En 2004, le W3C publie un nouvel ensemble de recommandations sur RDF pour remplacer celles de 1999.
- Pour des raisons de compatibilité avec l'existant, certains aspects sont conservés malgré les débats qu'ils suscitent, mais les défauts considérés comme majeurs sont corrigés.
- Après cet échec relatif, l'appellation *Semantic Web* tombe peu a peu en disgrâce. Certains défenseurs de RDF parlent plus modestement de *Data Web*, puis de *Web of Linked Data* (2006).

Fig. 2.2 – source : http://www.w3.org/RDF/icons/

Syntaxe(s) et sémantique

RDF 2004 définit :
- une syntaxe abstraite (modèle de donnée),
- une sémantique pour *interpréter* la syntaxe abstraite,
- plusieurs syntaxes concrètes pour représenter/échanger la syntaxe abstraite.

2.2 Syntaxe abstraite

2.2.1 Triplet

Toute information en RDF est représentée par un *triplet* :

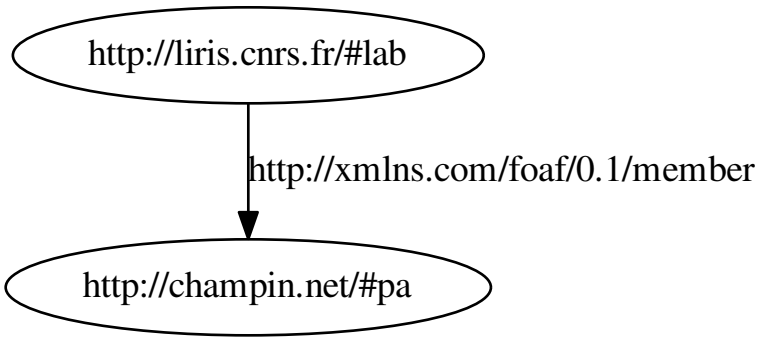
Le laboratoire LIRIS (sujet)
a pour membre (prédicat)
Pierre-Antoine Champin (objet)

Nommage

Les choses sont nommées par des URIs :

http://liris.cnrs.fr/#lab
http://xmlns.com/foaf/0.1/member
http://champin.net/#pa

On peut représenter ceci graphiquement :



Remarque

Dans RDF, les URIs ne sont utilisés que comme des **identifiants opaques**.

Les représentations qui sont éventuellement accessibles via ces URIs (par *déréférencement*) n'ont aucune influence sur leur sémantique.

Notons cependant que le mouvement Linked data préconise une utilisation *particulière* des URIs dans laquelle les réprésentations doivent être cohérentes avec la sémantique.

2.2.2 Préfixes

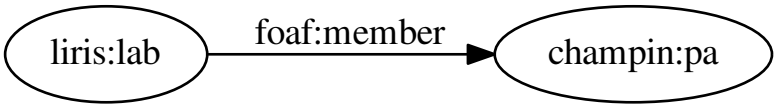
Pour simplifier les **notations**, on définit des préfixes courts correspondant à des préfixes d'URI :

liris : → `http://liris.cnrs.fr/#`
foaf : → `http://xmlns.com/foaf/0.1/`
champin : → `http://champin.net/#`

On utilise ensuite des *noms préfixés* :

liris :lab foaf :member champin :pa

et également sous forme graphique :

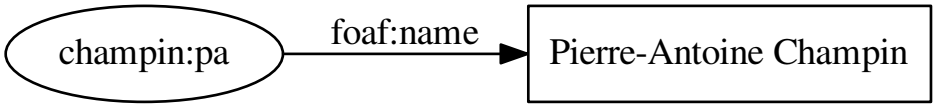


2.2.3 Littéraux

On peut également lier une ressource à une *donnée typée* (chaîne de caractère, entier, réel...), nommée un littéral.

champin :pa foaf :name "Pierre-Antoine Champin"

Traditionnellement, on représente les littéraux par des nœuds rectangulaires :

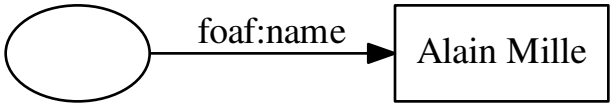


2.2.4 Nœuds vierges

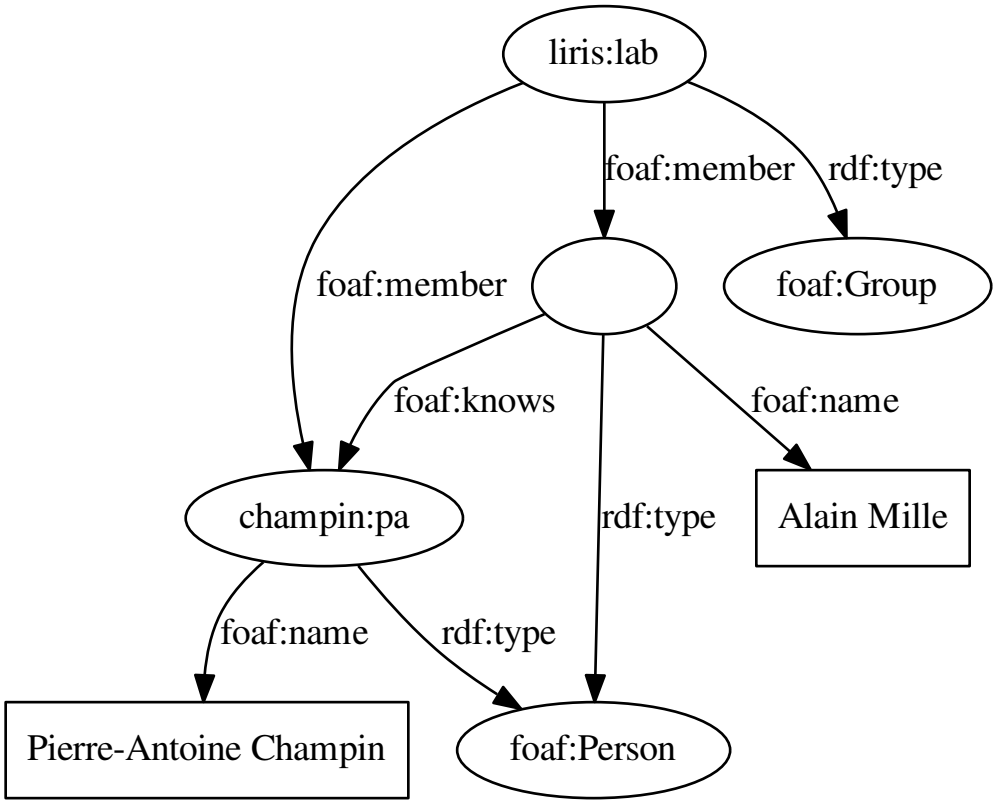
Enfin, RDF permet de parler d'une ressource sans connaître son URI. Cela revient en logique à utiliser une variable quantifiée existentiellement.

(quelque chose) foaf :name "Alain Mille"

Graphiquement, on représente cette ressource par un nœud vierge (*blank node*).



2.2.5 Exemple de graphe



2.3 Sémantique

2.3.1 Principe général

RDF est muni d'une sémantique en théorie des modèles, ou sémantique dénotationnelle :

- chaque nœud du graphe dénote une ressource (NB : les littéraux sont considéré comme des ressources particulières),
- chaque ressource peut être associée à une relation binaire entre ressources,
- chaque arc signifie que les ressources dénotés par les nœuds vérifient la relation associée à (la ressource dénotée par) l'URI de l'arc

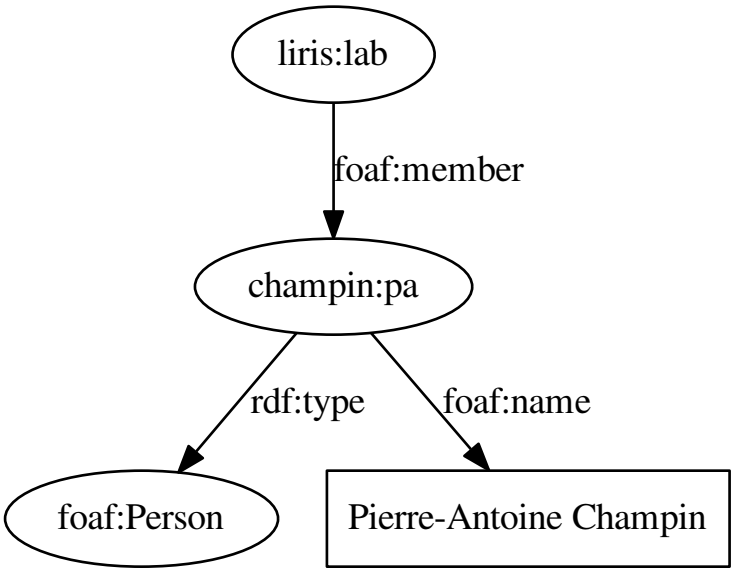
2.3.2 Inférences

De cette sémantique découlent les inférences que l'ont peut faire sur un graphe RDF (indépendamment de la sémantique des URIs utilisés dans ce graphe).

- Par analogie, les seules inférences que l'on puisse faire sur un arbre XML indépendamment des termes utilisés sont de changer l'ordre des attributs, et de changer les préfixes (si l'on considère les espaces de noms).

Monotonie

Étant donné un graphe, tout sous-graphe en est une conséquence.



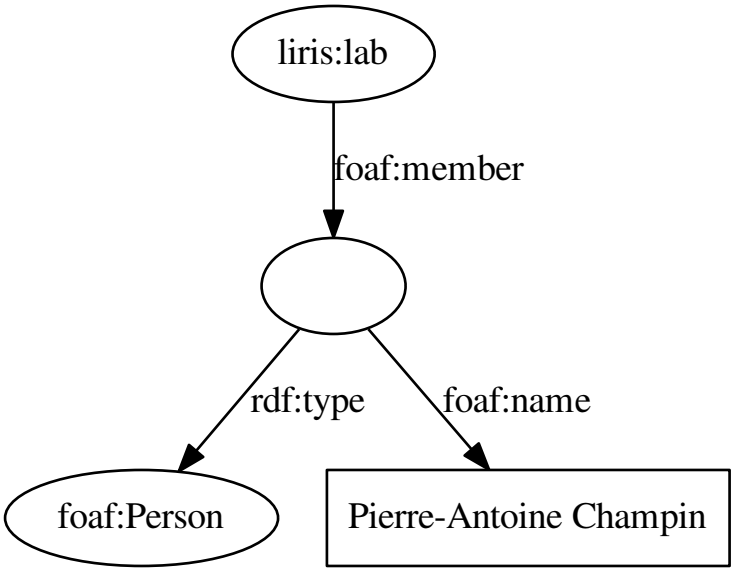
Monotonie (suite)

Aucune conclusion que l'on peut tirer d'un graphe RDF ne peut pas être contredite par des informations présentes ailleurs sur le Web.

→ On est donc toujours en droit de *fusionner* des graphes RDF, même provenant de sources indépendantes

Anonymisation

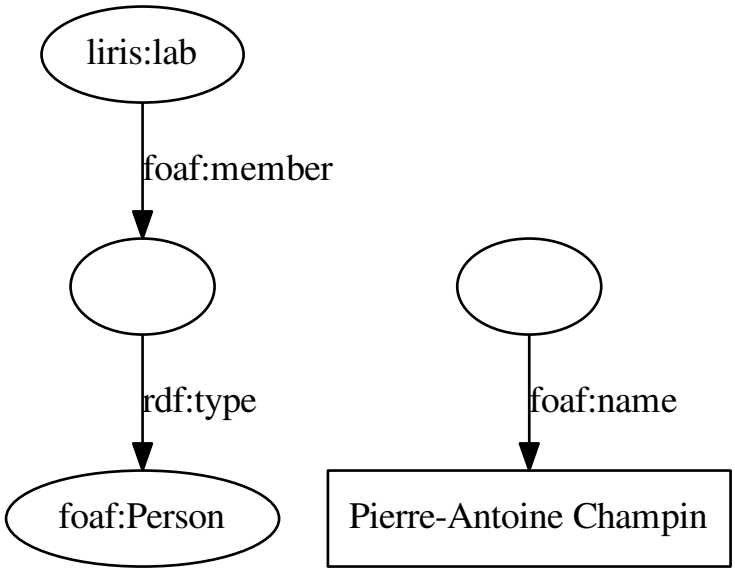
Tout nœud peut être remplacé par un nœud vierge :



(rappel : les nœuds vierges s'interpètent comme des variables quantifiées existentiellement)

Éclatement

Tout nœud vierge peut être éclaté en plusieurs nœuds vierges :



2.3.3 Sémantique additionelle

Bien sûr, des inférences supplémentaires peuvent être faites en prêtant une sémantique particulière aux URIs utilisés dans le graphe,

— par exemple en décidant qu'une relation binaire est symétrique ou transtitive.

Cependant, cette sémantique devra respecter la sémantique dénotationnelle de RDF ; notamment, les inférences supplémentaires ne devront *pas* contredire les inférences précédente (monotonie) :

— sans quoi les outils ne connaissant pas cette sémantique particulière tireront des conclusions erronées.

Sémantique additionelle (suite)

On verra plus tard des langages (RDF-Schema, OWL) permettant de définit la sémantique de certains URIs.

Analogie : lorsqu'on définit un format XML, on prête une sémantique particulière aux éléments et attributs de ce format, mais on ne peut *pas* prêter de sémantique à l'ordre des attributs ;

— sémantiquement, ce ne serait plus du XML,

— pragmatiquement, les outils standards (analyseur syntaxique, sérialiseurs) ne permettraient pas de contrôler cet aspect de la syntaxe.

2.4 Syntaxes concrètes

2.4.1 RDF/XML

— première syntaxe recommandée par le W3C (1999)

— basée sur XML

— relativement complexe et verbeuse

Syntaxe http://www.w3.org/TR/rdf-syntax-grammar/

Valideur http://www.w3.org/RDF/Validator/

RDF/XML : exemple

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"                    >
 <foaf:Group rdf:about="http://liris.cnrs.fr/#lab">
  <foaf:member>
   <foaf:Person>
    <foaf:name>Alain Mille</foaf:name>
    <foaf:knows
     rdf:resource="http://champin.net/#pa"/>
   </foaf:Person>
  </foaf:member>
  <foaf:member>
   <foaf:Person rdf:about="http://champin.net/#pa">
    <foaf:name>Pierre-Antoine Champin</foaf:name>
   </foaf:Person>
  </foaf:member>
  </foaf:Group>
</rdf:RDF>
```

2.4.2 Turtle : Terse RDF Triple Language

— dérivée du langage N3
— recommandé par le W3C en 2014
— vise la simplicité et la compacité

Syntaxe http://www.w3.org/TR/turtle/

Turtle : exemple

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix champin: <http://champin.net/#> .

liris:lab
    a foaf:Group ;
    foaf:member champin:pa, _:am .
champin:pa
    a foaf:Person ;
    foaf:name "Pierre-Antoine Champin" .
_:am
    a foaf:Person ;
    foaf:name "Alain Mille" ;
    foaf:knows champin:pa .
```

Turtle : exemple 2

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix champin: <http://champin.net/#> .

liris:lab
```

```
    a foaf:Group ;
    foaf:member champin:pa, [
      a foaf:Person ;
      foaf:name "Alain Mille" ;
      foaf:knows champin:pa .
    ] .
champin:pa
    a foaf:Person ;
    foaf:name "Pierre-Antoine Champin" .
```

2.4.3 RDFa : RDF in attributes

RDFa est une utilisation d'attributs (existants ou supplémentaires) de (X)HTML pour y inclure du RDF (à la manière des micro-formats) :

— facilite la migration de contenus HTML vers RDF
— facilite la maintenance en cohérence de la version HTML et des données RDF (*DRY : Don't Repeat Yourself*)

Syntaxe http://www.w3.org/TR/rdfa-primer/

Valideur http://check.rdfa.info/

Distiller http://www.w3.org/2012/pyRdfa/

RDFa : exemple

```
<p prefix="foaf http://xmlns.com/foaf/0.1/"
   about="_:am">
  <span property="foaf:name"
    >Alain Mille</span>
  est membre du
  <a rev="foaf:member" href="http://liris.cnrs.fr/#lab"
    >LIRIS</a>.
  Il connaît
  <span rel="foaf:knows" href="http://champin.net/#pa">
    <span property="foaf:name"
      >Pierre-Antoine Champin</span>,
    un autre membre de
    <span rev="foaf:member" href="http://liris.cnrs.fr/#lab">
      ce laboratoire.</span>
  </span> </p>
```

2.4.4 JSON-LD

— Rappel : JSON est un langage d'échange de données, basé sur Javascript, et très utilisé en développement web.
— JSON-LD (JSON Linked Data) permet d'interpreter une structure JSON comme du RDF,
— grâce à un *contexte* (implicite ou explicite).
— Objectif : faciliter l'adoption de RDF (syntaxe abstraite) auprès des développeurs d'applications web.

Syntaxe http://www.w3.org/TR/json-ld-syntax/

Valideur http://json-ld.org/playground/

JSON-LD : exemple

```
{ "@context" : { /* ... */                                    },
 "@id": "http://liris.cnrs.fr/#lab",
 "@type": "Group",
 "member": [
   {
     "@id": "http://champin.net/#pa",
     "@type": "Person",
     "name": "Pierre-Antoine Champin"
   },
   {
     "@type": "Person",
     "name": "Alain Mille",
     "knows": "http://champin.net/#pa"
   }
 ]
}
```

2.4.5 Autres syntaxes

- Comme l'illustrent RDFa et JSON-LD, tout langage peut être interprété comme du RDF :
  - dialectes en XML (GRDDL)
  - microformats (http://http://microformats.org/)
  - microdata (http://www.data-vocabulary.org/)
- ← Prépondérance de la syntaxe abstraite.
- Difficulté : faire correspondre des URIs là ou d'autres langages utilisent des termes « locaux ».

2.5 Dataset et requêtes

2.5.1 Dataset

Lorsqu'on stocke des données en RDF, il est parfois nécessaire de distinguer plusieurs graphes.

Un *dataset* RDF est constitué :
- d'un graphe par défaut,
- d'un ensemble de couples <URI, Graphe>
  /!\ aucune relation sémantique imposée entre l'URI et le graphe

2.5.2 SPARQL

SPARQL est un ensemble de recommandations définissant :
- un langage d'interrogation d'un *dataset* (SPARQL)
- un langage de mise à jour d'un *dataset* (SPARQL-update ou SPARUL)
- un protocole REST-like pour envoyer des requêtes SPARQL à un *dataset*

Référence http://w3.org/sparql

Tutoriel http://jena.sourceforge.net/ARQ/Tutorial/

SPARQL : exemple 1

« Trouver les membres du LIRIS (URI et nom) que je connaît ».

```
PREFIX foaf: <http://xmlns.com/foaf/0.1.> #...

SELECT ?p ?n
WHERE {
  champin:pa foaf:knows ?p .
  liris:lab foaf:member ?p .
  ?p foaf:name ?n .
}
```

SPARQL : exemple 2

« Construire l'union des graphes créés par Pierre-Antoine Champin ».

```
PREFIX dct: <http://purl.org/dc/terms/>

CONSTRUCT { ?s ?p ?o }
WHERE {
  ?g dct:creator <http://champin.net/#pa> .
  GRAPH ?g { ?s ?p ?o }
}
```

/!\ pré-supposition sur la sémantique des URIs des graphes

SPARUL : exemple

« Remplacer `foaf:firstName` par `foaf:givenName` pour tous les membres du LIRIS ».

```
PREFIX foaf: <http://xmlns.com/foaf/0.1.>

DELETE {
  ?m foaf:firstName ?n .
}
INSERT {
  ?m foaf:givenName ?n .
}
WHERE {
  <http://liris.cnrs.fr/#lab> foaf:member ?m .
  ?m foaf:givenName ?n .
}
```

CHAPITRE 3

Vocabulaires et ontologies

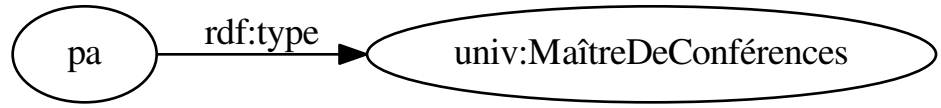
author Pierre-Antoine Champin

3.1 Introduction

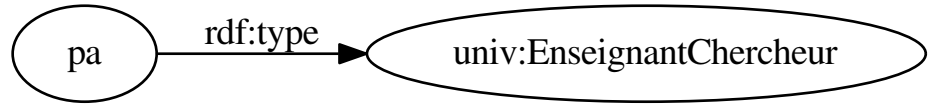
3.1.1 Motivation

Objectif : expliciter *formellement* la sémantique des vocabulaires (en conformité avec la sémantique de RDF), afin de
- limiter les problèmes d'ambigüité sur les termes
- permettre leur découverte dynamique
  - relations sémantiques internes
- assurer l'interopérabilité
  - relations sémantiques avec d'autres vocabulaires

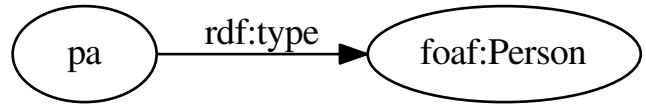
3.1.2 Exemples d'inférences



$\models_{KB}$



$\models_{KB}$



3.1.3 Expression de la sémantique

On souhaite bien sûr exprimer la sémantique des termes en utilisant RDF lui même. Il faut donc définir un *méta-vocabulaire*, dont la sémantique soit connue *a priori*.

— Analogie : XML-Schema est un vocabulaire XML dont la sémantique est connue *a priori*, et qui permet d'exprimer la structure de nouveaux vocabulaires.

3.2 RDF-Schema

3.2.1 Présentation

— RDF-Schema (ou RDF-S) est une recommandation du W3C publiée en même temps que RDF (1999 et révisée en 2004).
— Il permet d'exprimer une hiérarchie de classes et une hiérarchie de propriétés (relations).
→ hiérarchie au sens *large* : treillis
— Il permet aussi d'exprimer des contraintes sémantiques sur les propriétés et les classes.
/!\ contrainte sémantique ≠ contrainte d'intégrité
— Le préfixe habituellement associé à ce méta-vocabulaire est `rdfs:`.

3.2.2 Sémantique

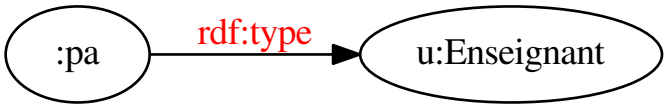
La sémantique de RDF-Schema est construite sur la sémantique de RDF :
— $\mathfrak{I}$ est la fonction d'interprétation qui associe une ressource à un nœud ;
— $\mathfrak{I}_p$ est la fonction qui associe une relation binaire à une ressource
On y ajoute une interprétation spécifique des classes :
— $\mathfrak{I}_c$ associe un ensemble de ressource à une ressource
Enfin, une interprétation conforme à RDF-S doit vérifier un certain nombre de contraintes qui traduisent la sémantique propre aux termes du méta-vocabulaire.

3.2.3 rdf :type

Contrainte :

$$(\mathfrak{I}(r), \mathfrak{I}(C)) \in \mathfrak{I}_p(\mathfrak{I}(\text{rdf :type}))$$
$$\equiv \mathfrak{I}(r) \in \mathfrak{I}_c(\mathfrak{I}(C))$$

Exemple :

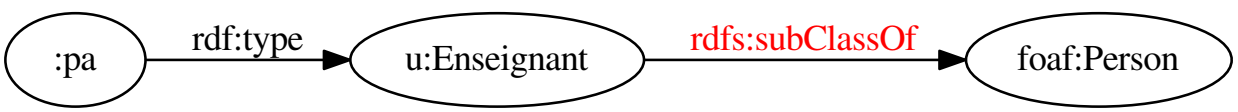


3.2.4 rdfs :subClassOf

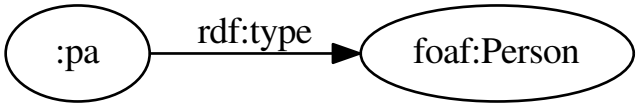
Contrainte :

$$(\mathfrak{I}(C), \mathfrak{I}(D)) \in \mathfrak{I}_p(\mathfrak{I}(\text{rdfs :subClassOf}))$$
$$\rightarrow \mathfrak{I}_c(\mathfrak{I}(C)) \subseteq \mathfrak{I}_c(\mathfrak{I}(D))$$

Exemple :



$\models$

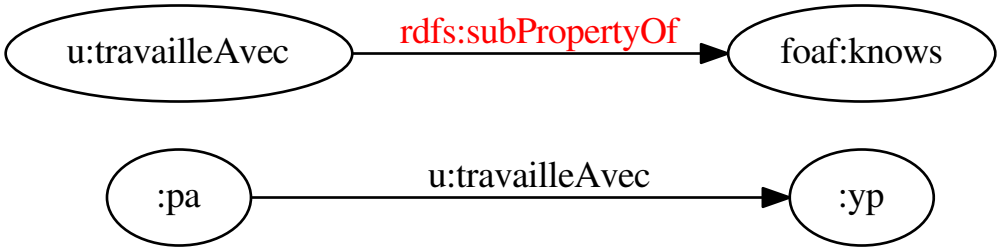


3.2.5 rdfs :subPropertyOf

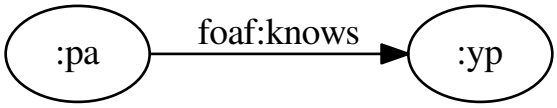
Contrainte :

$$(\mathfrak{I}(p), \mathfrak{I}(q)) \in \mathfrak{I}_p(\mathfrak{I}(\text{rdfs :subPropertyOf}))$$
$$\rightarrow \mathfrak{I}_p(\mathfrak{I}(p)) \subseteq \mathfrak{I}_p(\mathfrak{I}(q))$$

Exemple :



$\models$

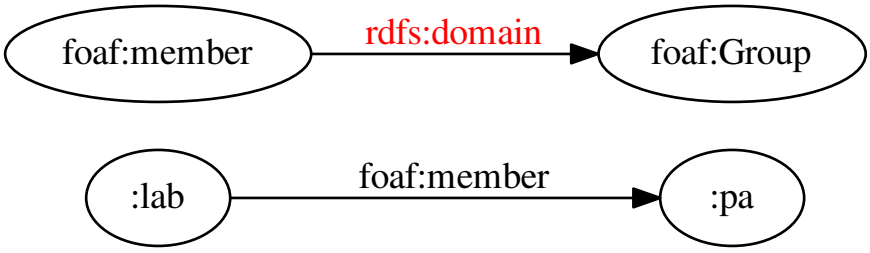


3.2.6 rdfs :domain

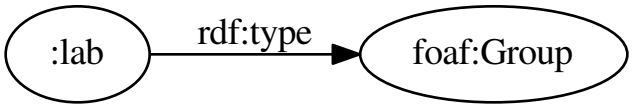
Contrainte :

$$(\mathfrak{I}(p), \mathfrak{I}(C)) \in \mathfrak{I}_p(\mathfrak{I}(\text{rdfs :domain}))$$
$$\wedge (r, s) \in \mathfrak{I}_p(\mathfrak{I}(p)) \rightarrow r \in \mathfrak{I}_c(\mathfrak{I}(C))$$

Exemple :



$\models$

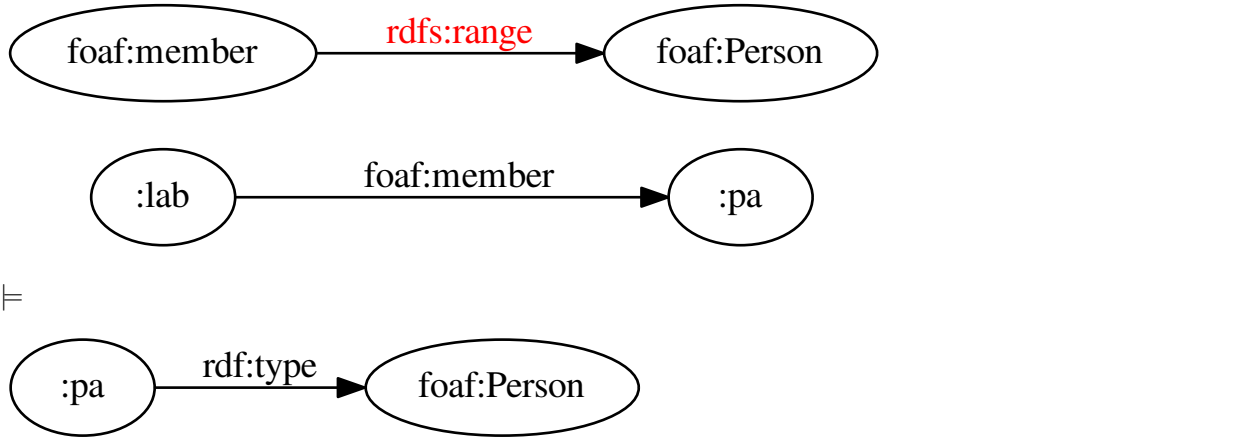


3.2.7 rdfs :range

Contrainte :

$$(\mathfrak{I}(p), \mathfrak{I}(C)) \in \mathfrak{I}_p(\mathfrak{I}(\text{rdfs :range}))$$
$$\wedge (r, s) \in \mathfrak{I}_p(\mathfrak{I}(p)) \rightarrow s \in \mathfrak{I}_c(\mathfrak{I}(C))$$

Exemple :



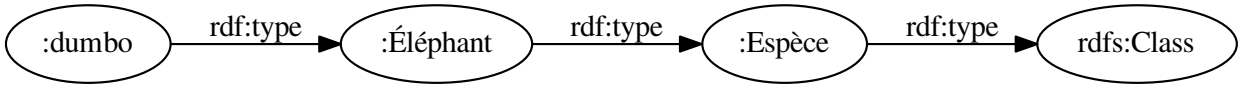
3.2.8 Documentation

RDF-Schema fournit aussi des termes pour *documenter* un vocabulaire :

— `rdfs:label` permet d'associer un libellé textuel à un URI (éventuellement plusieurs, par exemple dans plusieurs langues) ;
— `rdfs:comment` permet d'associer un commentaire textuel plus long ;
— `rdfs:seeAlso` permet de pointer vers une autre ressource.

3.2.9 Méta-modélisation

Rien n'empêche, en RDF-S, d'avoir une classe qui soit elle même une instance d'une autre classe (méta-classe). C'est d'ailleurs de cette manière que les classes sont identifiées.

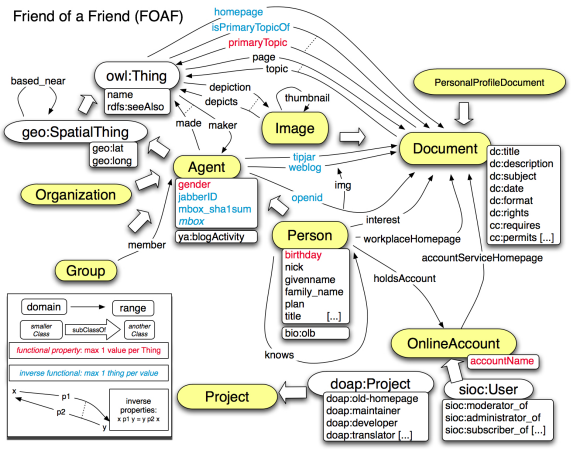


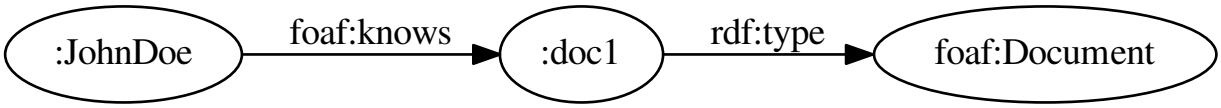
3.2.10 Exemple : FOAF

3.2.11 Contrainte sémantique ≠ contraintes d'intégrité

Les méta-propriétés `rdfs:domain` et `rdfs:range` ne servent *pas* à vérifier qu'un graphe serait « valide ». Il ne permettent que d'*inférer* des faits supplémentaires.

— Comme RDF-S n'a pas de négation, ceci n'entraîne jamais d'incohérence *formelle* au niveau des classes et des propriétés.
→ en d'autre termes, la sémantique de RDF-S ne permet pas de détecter les incohérences (conceptuelles) que pourraient entrainer ces inférences.





3.3 OWL

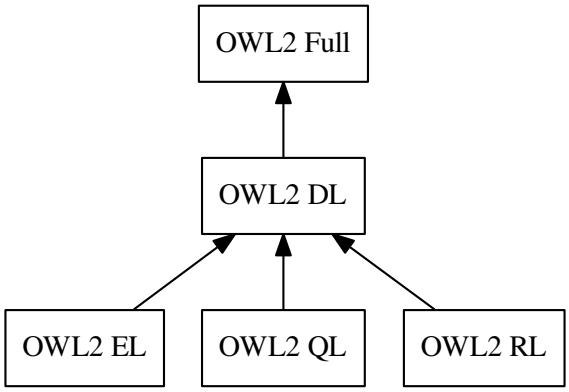
3.3.1 Présentation

OWL (Web Ontology Language) a été recommandé par le W3C en 2004, et sa version 2 en 2009.

— C'est un méta-vocabulaire (comme RDF-S) inspiré des **logiques de descriptions** avec valeurs concrètes (littéraux).
— Il définit plusieurs *profils* offrant des compromis différents en terme d'expressivité et de complexité.
— Il mime les capacités de méta-modélisation de RDF-S (*punning*).

3.3.2 Profils OWL 2

Full	aucune contrainte, indécidable
DL	minimum de contraintes, décidable mais très complexe
EL	quantifications existentielles (EL++), expressivité adaptée à certains domaines (biologie)
QL	peut s'implémenter au dessus d'un langage de requêtes (SQL)
RL	peut s'implémenter au dessus d'un langage à base de règles (Prolog)



3.3.3 Axiomes OWL

OWL offre des axiomes « de haut niveau » qui visent à
- améliorer la lisibilité de la base de connaissance
- optimiser le raisonnement
- contraindre l'utilisation de certains constructeurs à certaines formes d'axiome selon les profils (*e.g.* Property Chain)

Sur les concepts

OWL	LD
`rdfs:subClassOf`	$C \sqsubseteq D$
`rdf:type`	C(a)
`owl:equivalentClass`	$C \sqsubseteq D$ et $D \sqsubseteq C$
`owl:disjointWith`, `owl:AllDisjointClasses`	$C \sqsubseteq \neg D$

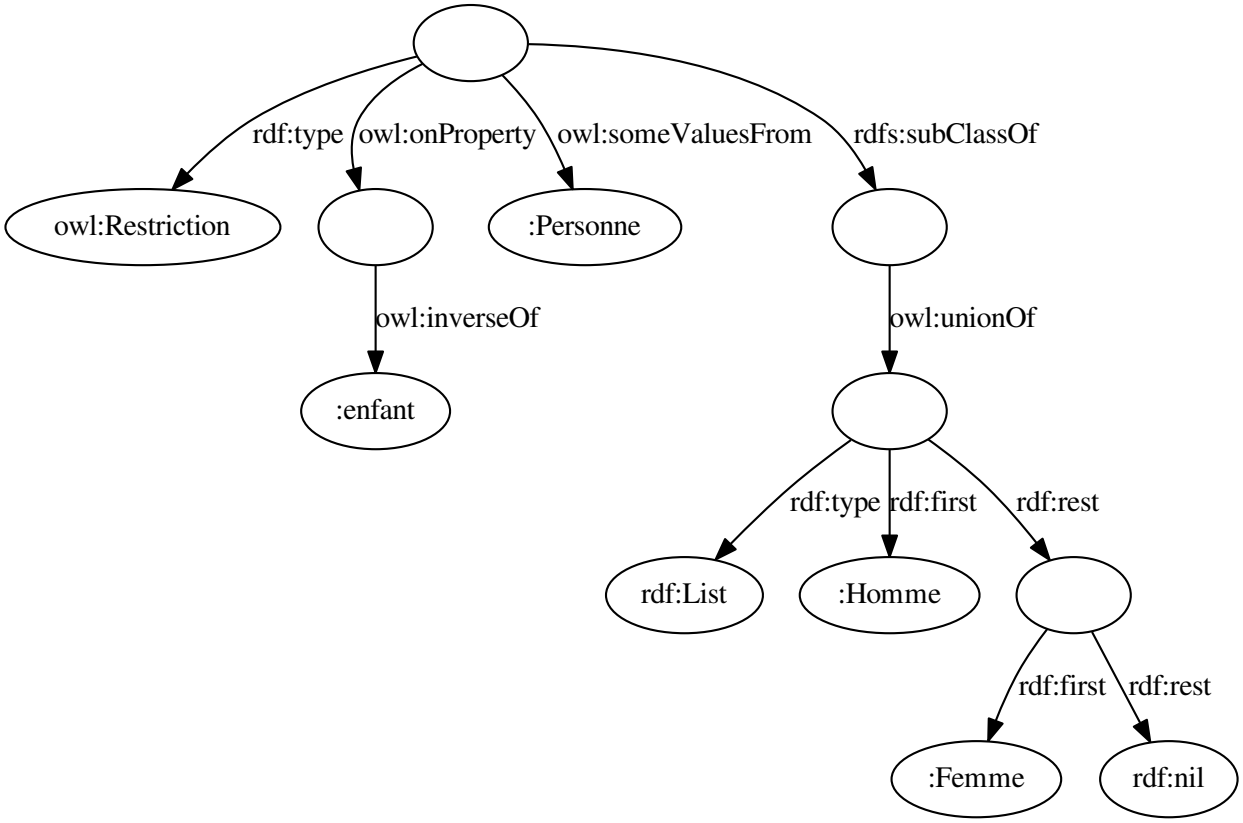
Sur les rôles

OWL	LD
`owl:FunctionalProperty`	$\top \sqsubseteq (\leq 1\ r)$
`owl:InverseFunctionalProperty`	$\top \sqsubseteq (\leq 1\ r^-)$
`owl:TransitiveProperty`	$r \circ r \sqsubseteq r$
`owl:SymetricProperty`	$r \sqsubseteq r^-$
`owl:AsymmetricProperty`	$r \sqsubseteq \neg(r^-)$
`owl:ReflexiveProperty`	$\top \sqsubseteq (\exists\ r\ self)$
`owl:IreflexiveProperty`	$\top \sqsubseteq \neg(\exists\ r\ self)$

Sur les rôles (suite)

OWL	LD
`rdfs:domain`	$\exists\ r \sqsubseteq C$
`rdfs:range`	$\exists\ r^- \sqsubseteq C$
`rdfs:subPropertyOf`	$r \sqsubseteq p$
`owl:equivalentProperty`	$r \sqsubseteq p$ et $p \sqsubseteq r$
`owl:inverseOf`	$r \sqsubseteq p^-$ et $p^- \sqsubseteq r$
`owl:propertyDisjointWith`, `owl:AllDisjointProperties`	$r \sqsubseteq \neg p$
`owl:propertyChainAxiom`	$p \circ q \circ ... \sqsubseteq r$

3.3.4 Représentation des LD en RDF



$\exists$ enfant$^-$ Personne $\sqsubseteq$ Homme $\sqcup$ Femme

CHAPITRE 4

Publication sur le web sémantique

author Pierre-Antoine Champin

4.1 Web sémantique et sémantique du web

4.1.1 Problème

Comment faire le lien entre
- la sémantique formelle des URIs en RDF/RDF-S/OWL, et
- leur sémantique « opérationnelle » sur le Web ?

→ nécessité d'établir de *bonnes pratiques*

4.1.2 Les quatres principes du Linked Data

- Utiliser des URIs pour nommer les choses (= ressources).
- Utiliser des URIs HTTP pour pouvoir obtenir des *représentations* de ces ressources.
- Fournit ces représentation en utilisant des langages et des protocoles standards (RDF, SPARQL).
- Inclure des liens pour permettre de découvrir de nouvelles ressources.

d'après Tim Berners-Lee, http://www.w3.org/DesignIssues/LinkedData.html

Remarques

On ne fait pas mention ici de sémantique, mais...
- un minimum de sémantique (RDF) est indispensable pour l'interopérabilité ;
- les vocabulaires peuvent être formalisés aussi précisément qu'on le veut (RDF-S, OWL).

4.2 Sémantique des URIs HTTP

4.2.1 Problème

- D'après les principes du *Linked Data*, tout objet d'intérêt devrait avoir un URI déréférenceable (en général `http:`)
  - y compris une personne, une organisation, un lieu, un concept...
- Or dans la sémantique de HTTP, une *ressource* est un objet *informatique*.

→ Ce problème est connu sous le nom httpRange-14

httpRange14

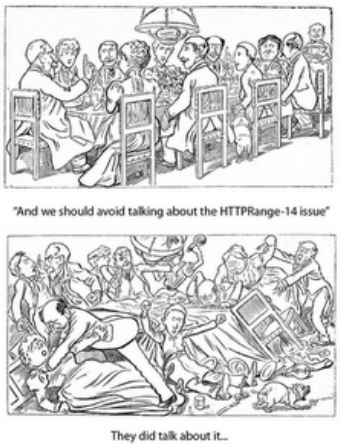


Fig. 4.1 – par Caran d'Ache et Ivan Herman

4.2.2 Solution 1 : identificateurs de fragment

— Un URI peut contenir un identificateur de fragment (*fragid*), commençant par #
— HTTP ignore les identificateurs de fragments (ils ne sont même pas envoyés au serveur)
  — donc la sémantique d'un URI contenant un *fragid* n'est pas contrainte par HTTP :
  — http://champin.net/ ≠ http://champin.net/#pa

Illustration

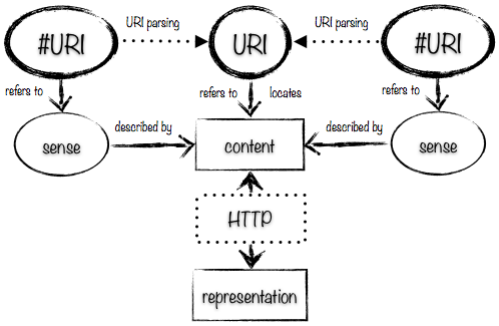


Fig. 4.2 – source : http://www.jenitennison.com/blog/node/170

Problèmes en suspens

— La sémantique du fragment est normalement spécifiée par le type de contenu, ce qui pose problème notamment pour la négociation de contenu (types de contenus multiples).
— Dans certains types de contenu, la sémantique du fragment est contrainte :

— XML, HTML → sous-arbre de l'arbre DOM
— ce qui pose problème pour RDFa...

4.2.3 Solution 2 : redirection

— Un code de retour `200 OK` signifie qu'on a affaire à une ressource au sens de HTTP (*information resource*)
  — Tout autre code n'engage pas HTTP sur la nature de la ressource.
— Si l'URI identifie un autre type de ressource (*non information resource*), il doit utiliser une redirection *via* `303 See Also`
  — ce qui peut impliquer de la négociation de contenu pour s'adapter au client.
  — exemple : http://dbpedia.org/resource/Lyon
  redirige vers http://dbpedia.org/data/Lyon (RDF)
  ou http://dbpedia.org/page/Lyon (HTML)

Illustration

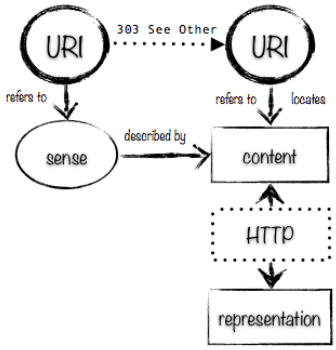


Fig. 4.3 – source : http://www.jenitennison.com/blog/node/170

4.3 Publier des données

4.3.1 Relational mappers

Outils pour exposer une base de données relationnelle en RDF , comme :
— un ensemble de pages RDF,
— un point d'accès SPARQL.

Nécessite une correspondance (*mapping*) entre les structures relationnelles (tables, colonnes) et la structure de graphe.

standard http://www.w3.org/2001/sw/rdb2rdf/

implémentation http://d2rq.org/

4.3.2 RDFa

— RDFa n'est pas une syntaxe comme les autres, puisqu'elle s'appuie sur HTML et transmet plus d'information que les triplets RDF
— RDFa peut être utilisé pour annoter du HTML généré manuellement ou automatiquement (CMS, Wiki, Blog...)

Exemple : Drupal

4.3.3 Wiki sémantique

- Principe : apporter
  - syntaxe simplifiée
  - ajout facile de contenu *et de liens*
  - travail collaboratif
  - formalisation incrémentale
- Plusieurs propositions
  - Semantic MediaWiki
  - Kiwi

CHAPITRE 5

Articles

— On Directly Mapping Relational Databases to RDF and OWL. Juan Sequeda, Marcelo Arenas, Daniel Miranker. WWW 2012.

— Template-based Question Answering Over RDF Data. Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Philipp Cimiano. WWW 2012.

— Holistic and Scalable Ontology Alignment for Linked Open Data. Toni Gruetze , Christoph Böhm , Felix Naumann. LDOW 2012.

— OWL : Yet to arrive on the Web of Data ?. Birte Glimm , Aidan Hogan , Markus Krötzsch , Axel Polleres. LDOW 2012.

— Concept-Based Semantic Difference in Expressive Description Logics. Rafael S. Gonçalves, Bijan Parsia and Ulrike Sattler. ISWC 2012.

— Hybrid SPARQL Queries : Fresh vs. Fast Results. Jürgen Umbrich, Marcel Karnstedt, Aidan Hogan and Josiane Xavier Parreira. ISWC 2012.

— Ontology Constraints in Incomplete and Complete Data. Peter Patel-Schneider and Enrico Franconi. ISWC 2012.

— SRBench : A Streaming RDF/SPARQL Benchmark. Ying Zhang, Minh-Duc Pham, Oscar Corcho and Jean Paul Calbimonte ISWC 2012.

— A publishing pipeline for Linked Government Data. Fadi Maali, Richard Cyganiak and Vassilios Peristeras. ESWC 2012.

— Graph Kernels for RDF data. Uta Lösch, Stephan Bloehdorn and Achim Rettinger. ESWC 2012

— Preserving Information Content in RDF using Bounded Homomorphisms. Audun Stolpe and Martin G. Skjæveland. ESWC 2012

— Exchange and Consumption of Huge RDF Data. Miguel A. Martinez-Prieto, Mario Arias Gallego and Javier D. Fernández. ESWC 2012