RDF Schema pour les ontologies légères

Bruit ≠ Précision

Agences I'RAM

La Galère
148, rue Victor Hugo
76600 Le Havre

L'Agence de la Presse et des Livres
38, rue Saint Dizier BP 445
54001 Nancy Cédex

Manqué ≠ Rappel

RESUME DU ROMAN DE VICTOR HUGO

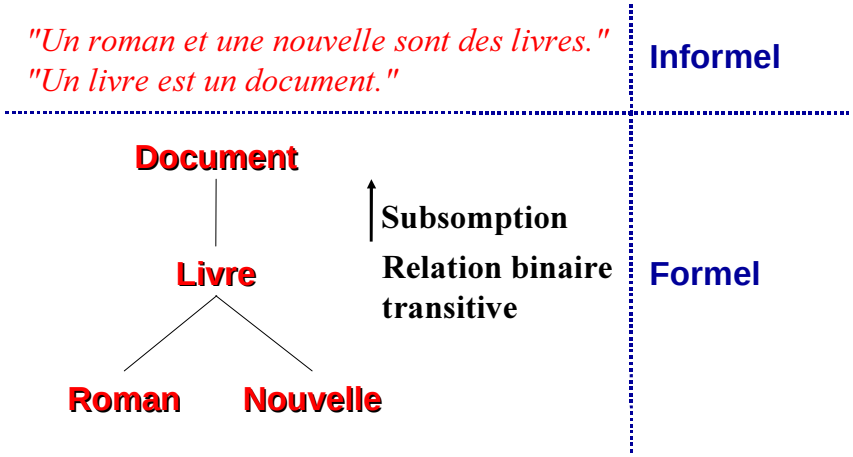
NOTRE DAME DE PARIS
(1831) - 5 parties

L'enlèvement . Livres 1-2 : 6 janvier 1482. L'effrayant bossu Quasimodo

Les livres de Hugo ?

- Réponse basée sur structuration des concepts:
  - objets / catégorie & identification
  - hiérarchie de catégories : structure d'abstraction spécialisation / généralisation
- Réponse basée sur un consensus (émetteur, public, récepteur)
- Cette structure et ce consensus sont ce que l'on appelle une 'ontologie'
  - Description de l'existant et de ses catégories exploitée dans des solutions informatiques
  - En informatique une "**o**ntologie" est un objet et non une discipline comme l'**O**ntologie en philosophie

- Manque une connaissance → identification
- Types de documents → acquisition
- Modéliser et formaliser → représentation



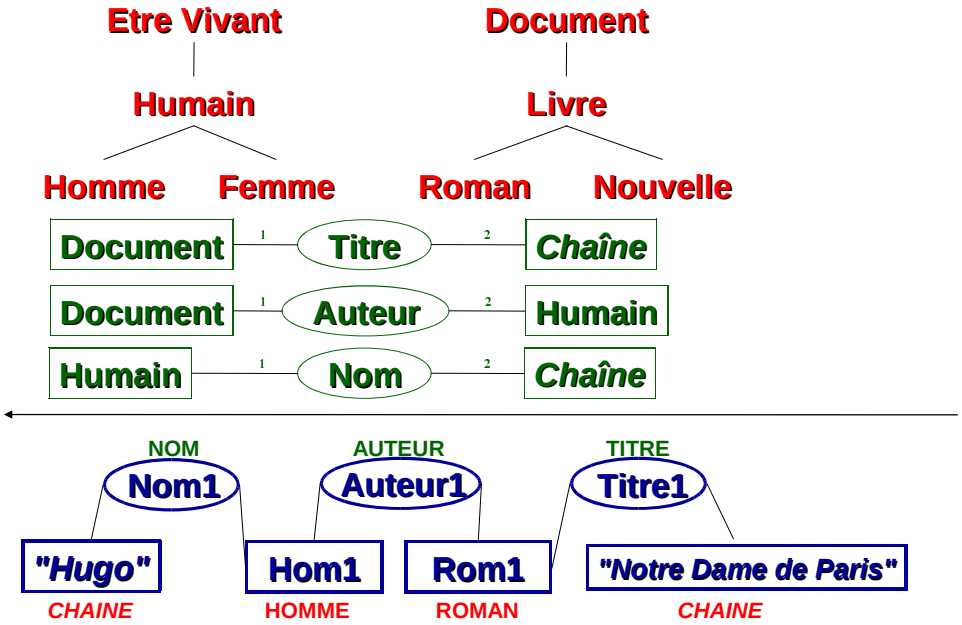
- Manque une connaissance → identification
- Types de documents → acquisition
- Modéliser et formaliser → représentation

"Un document a un titre.
Un titre est une chaîne de caractères"

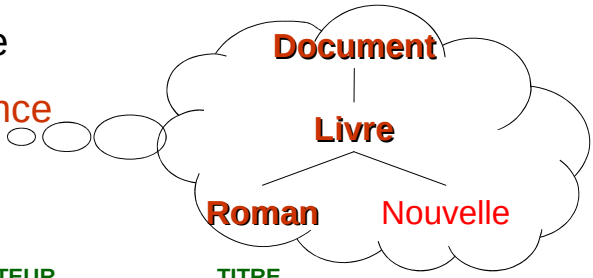
Informel

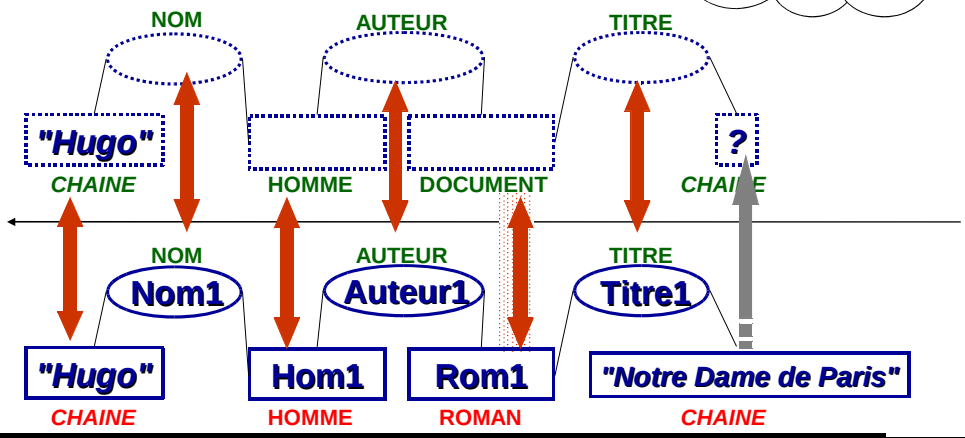
Formel

Hugo est l'auteur de Notre Dame de Paris

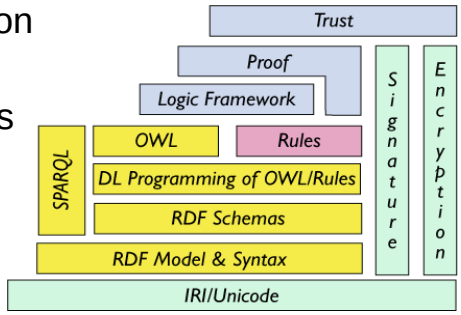


- Recherche : Requête
- Projection → Inférence
- Précision & Rappel





- Les 4 principaux standards du Web sémantique
  - RDF: un modèle de triplets pour décrire et connecter des ressources anonymes ou identifiées par un URI `(sujet, prédicat, objet)` / graphe orienté étiqueté
  - SPARQL: un langage de requête sur les graphes RDF
  - RDFS est un langage de déclarations et descriptions légères; typage des ressources et de leurs relations `subClassOf, subPropertyOf, range, domain`
  - OWL: 3 couches d'extension de l'expressivité (logique)
  - Un modèle en couche dans une direction d'extension; RDF sans RDFS, RDFS sans OWL, …



- 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 et donner leur signature
  - Liens hiérarchiques entre classes **et entre propriétés**
  - *Donner un URI aux concepts qui vous sont importants*

- Proche mais différent des modèles objets: propriétés en dehors des classes, multi-instanciation, héritage multiple classes et propriétés
- Squelette taxonomique d'une ontologie

- OWL sur une restriction de RDF/S
  - OWL Lite / DL / Full
  - Logiques de description
  - Vérification, classification, identification

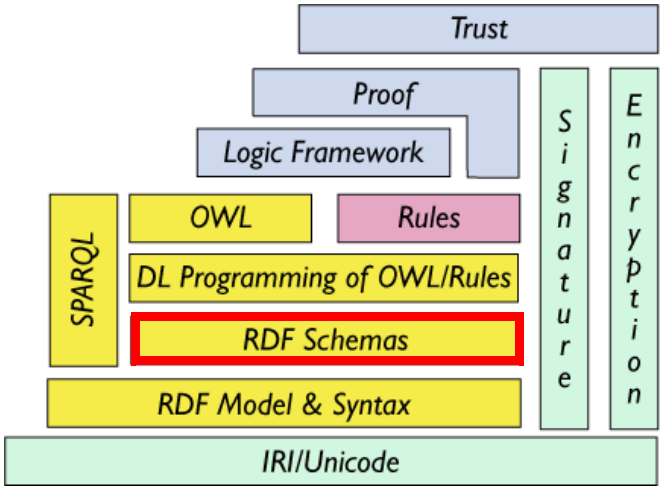
- Définition de classes (énumération, union, intersection, complément, disjonction, restriction valeur et cardinalité des propriétés)

- Caractérisation des propriétés (symétrique, transitive, fonctionnelle, inversement fonctionnelle, inverse)

- Gestion des équivalences, versions, documentations

- Un certain nombre d'outils/implémentations déjà disponibles (http://esw.w3.org/topic/SemanticWebTools):
  - C, C++, Java, PHP, Javascript, Python, Perles, C#, Ruby, Prolog, …
  - Plus de 17 Triple Stores (Jena, Oracle Spatial 10g, etc.)
  - Plus de 28 outils de développement (Altova, Top Quandrant, Protégé, etc.)
  - Beaucoup de livres: http://esw.w3.org/topic/SwBooks
- Les schémas ne sont pas forcément à créer mais parfois à traduire (ex: normes) ou parfois même déjà disponibles (ex: INSEE)

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



W3C, T Berners-Lee, Ivan Herman

- 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;

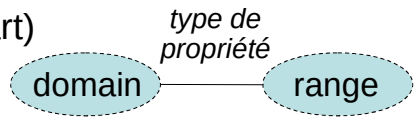
… des propriétés qui représentent des types de relations, des ensembles de relations possibles entre les ressources.

- 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;

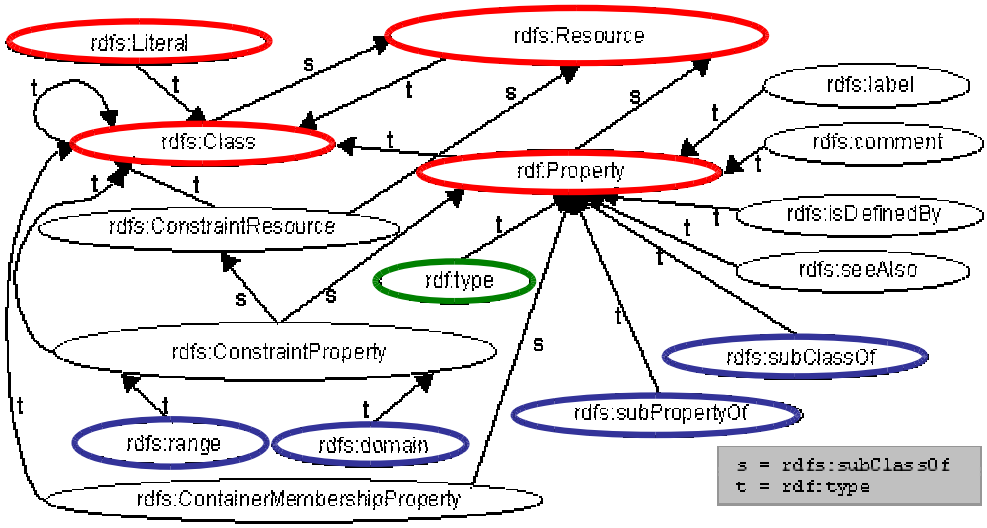
… la relation de sous-type (subsomption) pour dire qu'une classe/propriété est sous classe /propriété d'une autre et que ses instances sont aussi instances de l'autre.

- 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)
  - Le *range* (où la relation arrive)



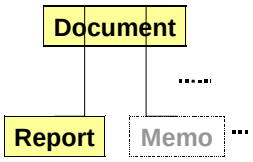
- Liens hiérarchiques des propriétés

- Propriétés : définies en dehors des classes
  - Modèles ouverts permettant à tout le monde de contribuer
  - Pas de raffinement ; pas de surcharge
- Multi-instanciation
  - Le typage multiple d'une même entité
  - Peut être vu comme des facettes
- Héritage multiple classes et propriétés
  - Deux hiérarchies de types: les classes, les propriétés
  - Chaque type peut hériter de zéro, un ou plusieurs types
- Inférences positives ≠ contraintes / vérification
  RDF/S est monotone, conjonctif et positif.

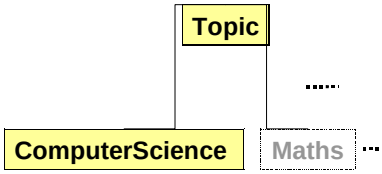


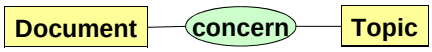
W3C http://www.w3.org/TR/2000/CR-rdf-schema-20000327/

```
class Document

class Report
  subClassOf Document
```

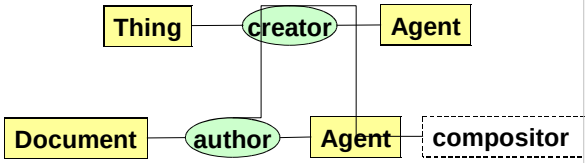



```
class Topic

class ComputerScience
  subClassOf Topic
```





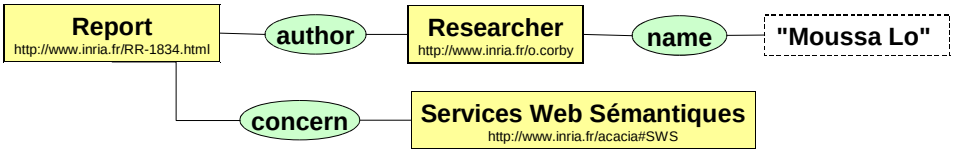

```
property concern
  domain Document
  range  Topic
```




```
property author
  domain Document
  range  Person
  subPropertyOf creator
```

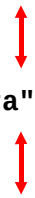
Le rapport RR-5663 a été écrit par le chercheur Moussa Lo et porte sur le sujet des Services Web Sémantiques

```
Report http://www.inria.fr/rrrt/rr-5663.html
   author urn://ugb.sn/mlo
   concern http://www.inria.fr/acacia#Java

Researcher urn://ugb.sn/mlo
   name “Moussa Lo”
```




```
 <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 for parent a
                                      female.</comment>
</rdf:Property>
```

```
<rdf:RDF 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.essi.fr/icws/2005-2006/humans.rdfs#"
xml:base="http://www.essi.fr/icws/2005-2006/humans.rdfs-
instances" >

<rdf:Description rdf:ID="Lucas">
 <rdfs:type rdf:resource="http://www.essi.fr/icws/2005-
2006/humans.rdfs#Man"/>
 <hasMother rdf:resource="#Laura"/>
</rdf:Description>

<Man rdf:ID="Lucas">
   <hasMother rdf:resource="#Laura"/>
</Man>


<rdf:Description rdf:ID="Lucas">
 <hasMother rdf:resource="#Laura"/>
</rdf:Description>

 <Man rdf:about="#Lucas" />
```



Exemple d'annotation 21

- URI pour les ressources annotées
  - URL de ressources web en ligne
  - URI de ressources abstraites ou physiques
- URI pour les types de ressources
  - URI pour identifier une classe, l'étendre, la spécialiser avec des sous-classes, etc.
  - URI pour typer une ressource
- URI pour les prédicats
  - URI pour identifier un type de propriété, l'étendre, la spécialiser avec des sous-relations
  - URI pour typer les liens entre les ressources

Importance des Uniform Ressource Identifiers 22

- 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>
```


- Les commentaires (comment) sont utilisés pour donner des définitions en langage naturel

```
<rdfs:Class rdf:about=‘#Woman’>
 <rdfs:subClassOf rdf:resource="#Person" />
 <rdfs:comment xml:lang=‘fr’>une personne adulte du sexe
  féminin</rdfs:comment>
 <rdfs:comment xml:lang=‘en’>a female adult
  person</rdfs:comment>
</rdfs:Class>
```

- Renvoi vers des notions connexes

```
<rdfs:Class rdf:about=‘#Man’>
 <rdfs:seeAlso rdf:resource=‘#Person’/>
</rdfs:Class>
```

- Si (c_2, subClassOf, c_1) et (x, type, c_2)
  alors (x, type, c_1)
  - *Exemple* : (Lo, type, Homme) ⇒ (Lo, type, Humain)
- Si (p_2, subPropertyOf, p_1) et (x, p_2, y)
  alors (x, p_1, y)
  - *Exemple* : (Lo, auteur, Note) ⇒ (Lo, créateur, Note)
- Si (c_3, subClassOf, c_2) et (c_2, subClassOf, c_1)
  alors (c_3, subClassOf, c_1) (transitivité)
- Si (p_3, subPropertyOf, p_2) et (p_2, subPropertyOf, p_1) alors (p_3, subPropertyOf, p_1) (transitivité)
- Idem réflexivité subClassOf et subPropertyOf

- Si (p, range, c) et (x, p, y) alors (y, type, c)
- Si (p, domain, c) et (x, p, y) alors (x, type, c)
  - *Exemple* : (aPourMere, range, Femme)
    (Fabien, aPourMere, Josette)
    ⇒ (Josette, type, Femme)
- Domain & Range sont optionnels (typage par défaut sur Resource)
- La signature est héritée
- Signature effective = conjonction des signatures héritées et spécifiées
- Sémantique de RDFS : http://www.w3.org/TR/rdf-mt/

- Un même objet vu sous plusieurs points de vue

```
<Man rdf:about="#John">
   <age>32</age>
   <name>smith</name>
</Man>

<Researcher rdf:about="#John">
   <subject>Math</subject>
   <rdf:type rdf:resource="Lecturer"/>
</Researcher>

<Goalkeeper rdf:about="#John"/>
```


Exercice

• Récupérer le fichier *human.rdfs* situé sur la page *http://www.u-picardie.fr/~furst/onto.html*

• Question 1 : Quel est l'espace de nommage associé à cette ontologie? Dans quels espaces de nommage sont définis les termes du langage RDF(S) : Class, Property, label, comment, range, domain, subClassOf, subPropertyOf, etc?

• Question 2 : Regardez le début du fichier et dessinez le sous graphe de la hiérarchie contenant les classes Animal, Man et Woman.

• Question 3 : Sur quelles classes porte la propriété âge?

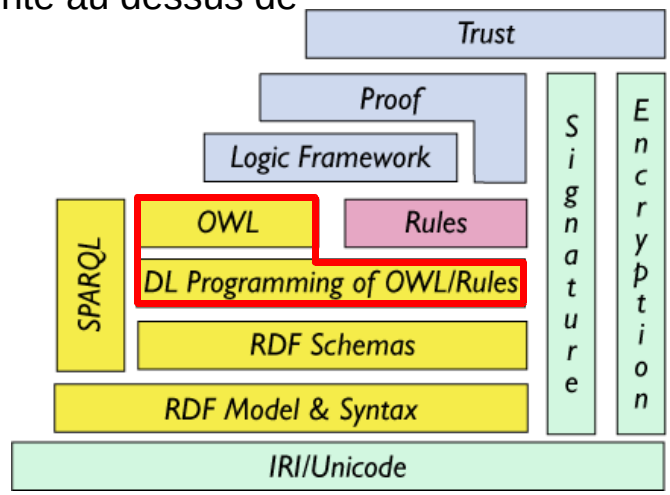
• Question 4 : quelles relations d'héritage existent entre les classes de l'ontologie?

• Exercice : donnez les inférences faites

```
c:creator rdfs:domain c:Person
i:Man241 c:creator i:Image262
i:Man241 rdf:type c:Person

c:author rdfs:subPropertyOf c:creator
c:author rdfs:range c:Document
i:Woman297 c:author i:Book812
i:Book812 rdf:type c:Document
i:Woman297 c:creator i:Book812
i:Woman297 rdf:type c:Person

c:aSoutenu rdfs:domain c:Docteur
c:aSoutenu rdfs:range c:These
i:Woman297 c:aSoutenu i:t127
i:Woman297 rdf:type c:Docteur
i:t127 rdf:type c:These

c:nbDeRoues rdfs:domain c:Vehicule
i:Man241 c:nbDeRoues "4"^^xsd:integer
i:Man241 rdf:type c:Vehicule
```

Ontologies lourdes en OWL

- OWL représenté au dessus de RDFS
  - OWL Lite
  - OWL DL
  - OWL Full



W3C, T Berners-Lee, Ivan Herman

- En réalité OWL est basé sur une restriction de RDF
  Classes ≠ Propriétés ≠ Individus
  Pas de modification du méta model

- Augmenter l'expressivité de la représentation des ontologies de RDFS :
  - Propriétés algébriques des relations
    ex: `ex:estMariéAvec` est symétrique.
  - Correspondances entre deux ontologies
    ex: `ex:Voiture` est équivalent à `ex:Car`
  - Contraintes de cohérence
    ex: `ex:Homme` est disjointe de `ex:Femme`
  - Définition formelles des classes
    ex: `ex:Manager(?x)` équivalent à `?x-(manage)-?y`
  - Restriction des propriétés et raffinement
    ex: pour `ex:Human` le range de `ex:child` est `ex:Human`

- OWL DL signifie OWL Description Logic
- Logiques de description séparation:
  - Concept / Rôle / Individu
  - Deux niveaux distincts:
    niveau terminologique: représentation et manipulation des concepts et des rôles (TBox) subsomption, hiérarchies de concepts et de rôles
    niveau factuel / assertionnel: description et manipulation des individus (ABox)
- Parallèle Concept ↔ Classe & Rôle ↔ Propriété
- Niveaux distincts: d'où la restriction de RDF/S

- Concept primitif (nommé) ou défini (définition formelle)
- Définition : description structurée (équations terminologiques)
- Les définitions utilisent des constructeurs pour donner:
  - les rôles associés au concept
  - les restrictions des rôles (co-domaine, cardinalité) valeurs de base / concepts

- le *et / and /* $\cap$ permet de définir une conjonction d'expressions conceptuelles
- Le *non / not /* $\neg$ correspond à la négation et ne porte que sur les concepts primitifs
- la quantification universelle *tout / all /* $\forall$ permet de préciser le co-domaine d'un rôle $\forall r.C$
- la quantification existentielle non typée *some / certains /* $\exists$ permet d'affirmer l'existence d'au moins un couple d'individus ($\exists r$) en relation r

primitifs

```
Personne ≤ Top
Ensemble ≤ Top
Homme ≤ Personne
Femme ≤ (and Personne (not Homme))
membre ≤ toprole
chef ≤ membre
```

incompatibles / disjointes

nécessaire

définis

```
Equipe = (and Ensemble
              (all membre Personne)
              (atleast 2 membre))

Petite-équipe = (and Equipe
                     (atmost 5 membre))

Equipe-moderne = (and Equipe
                      (atmost 4 membre)
                      (atleast 1 chef)
                      (all chef femme))
```

nécessaire & suffisant → classification

- Niveau factuel :

```
Equipe-moderne(Dreamteam)
Homme(Robert)
Personne(Roberta)
membre(Dreamteam, Robert)
membre(Dreamteam, Roberta)
membre(Dreamteam, Jules)
chef(Dreamteam, Roberta)
(atmost 4 membre) (Dreamteam)
```

- Inférences :
  - Dreamteam est une petite équipe (et une équipe)
  - Robert et Jules sont des personnes
  - Roberta est une Femme

- Test de subsomption: vérifier qu'un concept en subsume un autre (utile pour valider une classification)
- Classification : placer un concept ou un rôle dans la hiérarchie (assistance à la construction et l'évolution des ontologies)
- Test de satisfiabilité: vérifier qu'un concept admet des instances (utile pour vérifier la cohérence)
- Identification : retrouver les concepts les plus spécifiques dont un individu est susceptible d'être une instance
- Beaucoup de travaux sur la complexité algorithmiques // différentes familles de langages → Influence sur OWL

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

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*

- Définition en extension d'une classe i.e. en énumérant tous ses membres (utile en particulier pour les domaines d'attributs)

```
<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>
```

- 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>
```


- Définition d'une classe complémentaire

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

- Imposer une disjonction

```
<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>
```

- 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>
```


- Cardinalité d'une propriété : nombre d'instances différentes d'une propriété *i.e.* nombre de fois où une même ressource est utilisée comme point de départ (domain) d'une propriété

- 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`**

- 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 ignorée dans les inférences, uniquement utilisées pour documenter ou pour des extensions hors des inférences DL

- Propriété symétrique, xRy ⇒ yRx
- Exemple : `<owl:SymmetricProperty rdf:ID="hasSpouse" />`
- Propriété transitive, xRy & yRz ⇒ xRz
- Exemple : `<owl:TransitiveProperty rdf:ID="hasAncestor" />`
- Propriété fonctionnelle, xRy & xRz ⇒ y=z
- Exemple : `<owl:FunctionalProperty rdf:ID="hasMother" />`
- Propriété inversement fonctionnelle, xRy & zRy ⇒ x=z
- Exemple :
```
<owl:InverseFunctionalProperty
rdf:ID="NumSSociale" />
```
- Deux propriétés inverses, $xR_1y <=> yR_2x$
- Exemple :
```
<rdf:Property rdf:ID="hasChild">
   <owl:inverseOf rdf:resource="#hasParent"/>
</rdf:Property>
```

- Classes équivalentes : **`owl:equivalentClass`**
- Propriétés équivalentes : **`owl:equivalentProperty`**
- Instances identiques ou différentes : **`owl:sameAs`**, **`owl:differentFrom`**
- 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`**

- OWL Full contient tout ce que l'on a mentionné mais OWL Full n'est pas décidable
- OWL DL (Description Logic) est une première restriction (décidable mais avec des algorithmes parfois exponentiels) qui
  - Sépare Class, Thing, ObjectProperty, DatatypeProperty
  - N'autorise pas rdfs:Class, extension méta-modèle, cardinalité sur propriété transitive
- OWL Lite est une seconde restriction (i.e. restriction de OWL DL) qui n'autorise pas:
  - Union
  - Cardinalité autre que 0 ou 1

Quelques liens utiles

- Validateurs OWL :
  - En ligne : *http://www.mygrid.org.uk/OWL/Validator*
  - Jena (opensource Java) *http://jena.sourceforge.net/*
- Raisonneurs OWL :
  - Jena
  - Pellet (opensource Java) *http://clarkparsia.com/pellet*
  - Fact (classifieur, service web) *http://www.cs.manchester.ac.uk/~horrocks/FaCT/*
  - Racer (devenu payant) *http://www.racer-systems.com/*
- API : *http://owlapi.sourceforge.net/*
- Editeur : Protégé *http://protege.stanford.edu/*

Résumé

- Intégration de données à l'échelle du Web
  - Web actuel : en langage naturel pour les humains
  - Web sémantique : idem + en langage formel pour les machines; évolution et non révolution
  - Metadonnée = donnée au dessus des données i.e. des données au dessus du Web actuel
- But : interopérabilité, automatisation, réutilisation

- Langages, modèles et formats pour échanger…
  - Structure et nommage: XML, Namespaces, URI
    Roman -> http://essi.fr/ontologie#roman
  - Modèles et ontologies: RDF/S & OWL
    essi:Roman(x) ⇒ essi:Livre(x)
  - Protocoles et requêtes: HTTP, SOAP, SPARQL
  - A venir: règles, web services sémantiques, sécurité, etc.
- Rendre explicite ce qui existe déjà mais est implicite:
  - Capter, ex: types de ressources, auteur, date
  - Exposer ex: structures des formats ex: jpg/mpg, doc/xsl
  - Plein d'outils ont ce potentiel

- Compréhension partagée de l'information
  - Entre les personnes
  - Entre les applications
  - Entre les personnes et les applications

- Dans le « Web sémantique » le Web est dans les URI
  http://www.siteMachin.fr , ftp://ftp.ouvaton.org ,
  mailto:tartempion@triffouillis.fr , tel:+33492387788 ,
  http://siteMachin.fr/ontologie#roman, etc.
  et on peut dire tout sur tout.

- En construction…

Exercice

Le site *http://dbpedia.org/* permet de fouiller Wikipedia en utilisant RDF+OWL et SPARQL. Un point d'accès SPARQL avec formulaire est disponible à l'adresse *http://dbpedia.org/snorql/*

• Question 6.1 : afficher toutes les classes qui héritent directement de owl:Thing

• Question 6.2 : afficher toutes les *owl:ObjectProperty* et *owl:DataTypeProperty* ayant pour domaine *http://dbpedia.org/ontology/Place*

• Question 6.3 : afficher toutes les *Place* et leur ville la plus proche

• Question 6.4 : afficher toutes les *Place* et leur type

Question 6.5 : afficher toutes les *Event* qui font partie d'un autre *Event* et qui ne sont pas des *MilitaryConflict*

• Question 6.6 : afficher les facultés soeurs (*sisterCollege*). Comment ajouter la symétrie de cette relation dans l'ontologie?

• Question 6.7 : Lister tous ceux qui ont influencé (*dbpedia2:influenced*) Orwell (*George_Orwell*). Lister tous ceux par qui Orwell a été influencé (*influencedBy*). Comment ajouter dans l'ontologie la propriété inverse entre les deux relations?

Vérifier que Orwell a influencé Camus (*Albert_Camus*) qui a influencé Pamuk (*Orhan_Pamuk*). Est ce qu'Orwell a influencé Pamuk? Comment ajouter dans l'ontologie la propriété de transitivité sur la relation *influenced*?