

Langage OWL et ontologies

I.Mougenot

UM

*HMIN*₂₁₈ 2021

Contenu du cours

Comment tirer parti de modèles de connaissances sur le web ?

- aller au delà de la richesse de la représentation permise par le langage RDFS
- langage OWL et construction d'ontologies/bases de connaissances
- adossement à une famille de formalismes logiques (logiques de description)

Positionnement Web de données

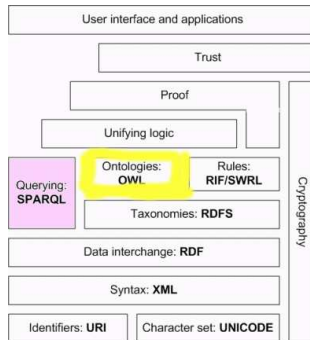


Figure: Rappel empilement de couches

Ontologie

Compréhension commune et formalisée, d'un domaine d'intérêt

- Essence de l'être [Aristote 350 Av. J.C.] (questionnement philosophique sur l'existence)
- Spécification formelle d'une conceptualisation partagée [T. Gruber 1993]
L'aspect formel assure une interprétation correcte de la connaissance modélisée, par les machines.
- Compréhension partagée d'un domaine d'intérêt à utiliser comme cadre unificateur pour résoudre tt problème de communication entre personnes et d'interopérabilité entre systèmes. [Uschold et Gruninger]

Mots clés : partage, interprétation

Ontologie

En informatique, a pris, pleinement, son essor avec le web sémantique

- concepts
- relations entre concepts
- axiomes (énoncés posés comme vrais, qui définissent des propriétés) portant sur les relations et sur les concepts

Bases de connaissances

Plusieurs significations selon le contexte

- base de faits uniquement
- base de faits + base d'énoncés véhiculant la connaissance (axiomes et/ou règles)

Ontologie OWL : deuxième vision : ABox (boîte des individus) + TBox (boîte des concepts et des propriétés)

Langage OWL (W3C)

Ontology Web Language : un standard du W3C

- OWL 2 ou OWL 2009
- relations entre concepts
- famille de langages de représentation de connaissances, pensée pour le web
- double adossement : standards du W3C (RDF et RDFS), logiques de description

Langage pour les ontologies : OWL 2

Enrichir l'expressivité des modèles (diagramme UML incomplet)

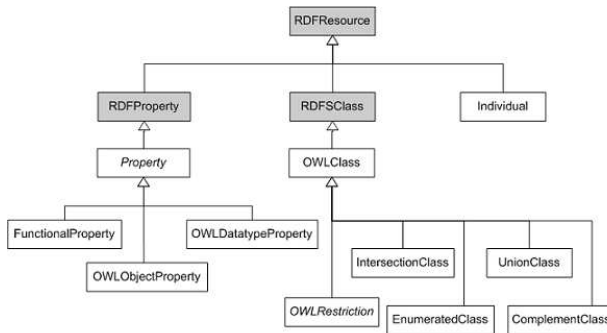


Figure: Articulation RDFS / OWL

primitives OWL 2

Les bases de la construction des classes OWL

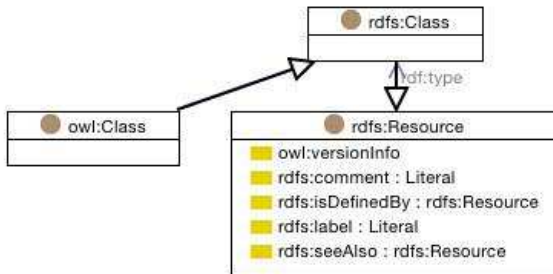


Figure: Extrait diagramme UML
(<https://projects.ics.forth.gr>)


primitives OWL 2

Synthèse visuelle des apports du langage

$\{a,b,c,d,e\}$ enumeration

 union

 \neq  disjunction


 symmetric

 transitive

 inverse

 intersection

 complement

 restriction

 cardinality

 equivalence

Figure: Figure empruntée à O. Corby INRIA Sophia Antipolis

Focus sur la transitivité

Notion de propriété transitive

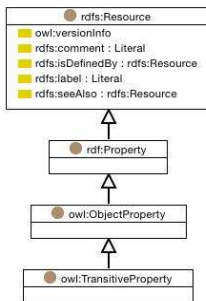


Figure: Extrait diagramme UML
(<https://projects.ics.forth.gr>)

Ontology Web Language (W3C) - non exhaustif

1 propriétés sur les classes

classes disjointes : Homme / Femme (owl:disjointWith)

classes équivalentes : Président / Chef de l'Etat
(owl:equivalentClass)

classes énumérées { Chat, Lion, Guépard, ... } (owl:oneOf)

2 propriétés sur les propriétés

propriété transitive (owl:TransitiveProperty)

propriété symétrique (owl:SymmetricProperty)

propriété inverse (owl:InverseOf)

3 type des propriétés

propriété caractéristique d'un concept (owl:DatatypeProperty)

propriété mettant en interaction deux concepts
(owl:ObjectProperty)

4 cardinalités, individus, négation, connecteurs logiques ...

Première illustration OWL

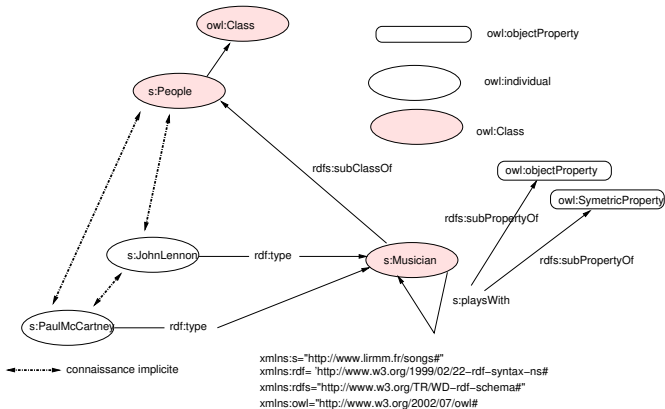


Figure: Aller vers des ontologies "riches"

Tom et Jerry (en RDFS)

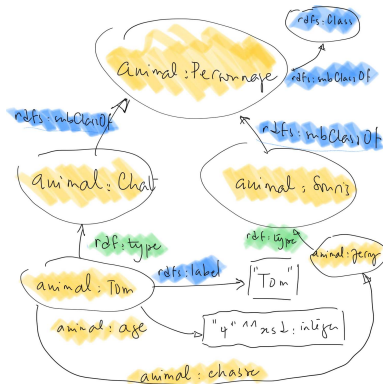


Figure: Retour sur l'exemple de Tom et Jerry

Tom et Jerry (OWL)

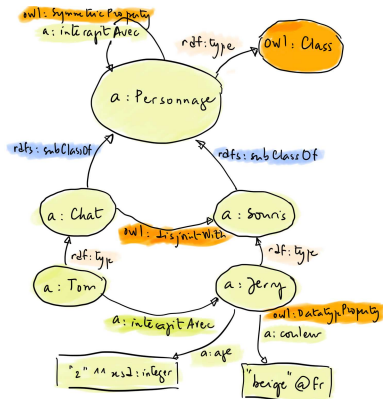


Figure: Faire évoluer l'exemple

Editeurs d'ontologies OWL

Faciliter la construction/manipulation d'ontologies

- Protégé <https://protege.stanford.edu/> (API OWL-API)
- SWOOP <http://www.mindswap.org/2004/SWOOP/>
- NeOn Toolkit <http://neon-toolkit.org/>
- TopBraid Composer
http://www.topquadrant.com/products/TB_Composer.html
- ...

voir la page https://www.w3.org/wiki/Ontology_editors

Construire l'ontologie à l'aide de Protégé

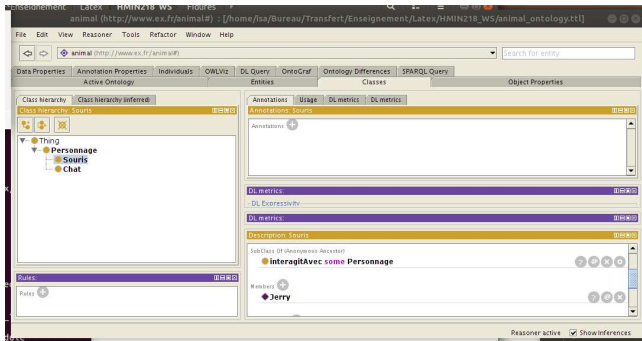


Figure: Editeur Protégé

Portion du graphe résultant

en Turtle

```
<http://www.ex.fr/animal#interagitAvec> rdf:type owl:ObjectProperty ,  
                                         owl:SymmetricProperty .  
  
<http://www.ex.fr/animal#Chat> rdf:type owl:Class ;  
  
                                rdfs:subClassOf <http://www.ex.fr/animal#Personnage> ;  
  
                                owl:disjointWith <http://www.ex.fr/animal#Souris> .  
  
<http://www.ex.fr/animal#Personnage> rdf:type owl:Class ;  
                                rdfs:subClassOf [ rdf:type owl:Restriction ;  
                                                owl:onProperty  
                                                  <http://www.ex.fr/animal#interagitAvec>  
                                                  ;  
                                                owl:someValuesFrom  
                                                  <http://www.ex.fr/animal#Personnage>  
                                              ] .
```

Listing 1: Tom et Jerry et OWL

Un personnage interagit avec au moins un autre personnage

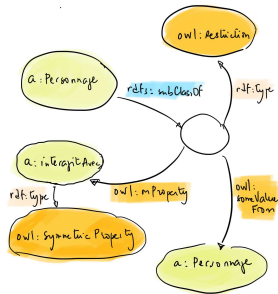


Figure: Restriction OWL et classe non-nommée

Explicitation de savoir implicite

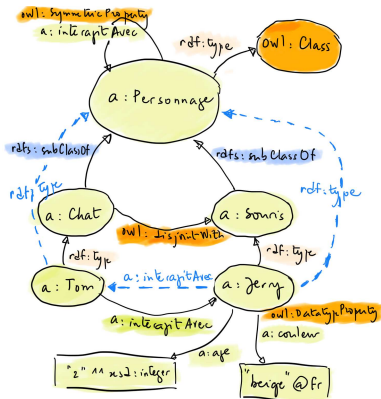


Figure: Fixer les interprétations

Fixer des interprétations au travers de OWL

Exploiter le typage des classes et des propriétés OWL

- 1 Transitivité (**owl:TransitiveProperty**) portant sur la propriété ancêtre
par ex. $\{ h:\text{Marie}, h:\text{ancetreDe}, h:\text{Jeanne} \}$ et $\{ h:\text{Jeanne}, h:\text{ancetreDe}, h:\text{Justine} \} \mapsto \{ h:\text{Marie}, h:\text{ancetreDe}, h:\text{Justine} \}$
- 2 Inverse (**owl:inverseOf**) portant sur les propriétés estParentDe et aPourParent
par ex. $\{ h:\text{Marie}, h:\text{estParentDe}, h:\text{Jeanne} \} \mapsto \{ h:\text{Jeanne}, h:\text{aPourParent}, h:\text{Justine} \}$
- 3 Classes équivalentes (**owl:EquivalentClass**) liant les classes Personne et Humain
par ex. $\{ h:\text{Personne}, \text{owl:EquivalentClass}, h:\text{Humain} \} \{ h:\text{pierre}, \text{rdf:type}, h:\text{Humain} \} \mapsto \{ h:\text{pierre}, \text{rdf:type}, h:\text{Personne} \}$

Quelques classifieurs disponibles

- internes à Jena (Apache)
- Pellet (Clark & Parsia)
- FaCT++ (Université de Manchester)
- HermiT (Information Systems Group)
- Hoolet (Université de Manchester)
- ...

Focus sur Pellet (OPenllet)

Ecrit en Java, pouvant être mobilisé sous forme d'API (avec Jena et OWL-API), en ligne de commande, ou comme plugin Protégé

- mécanismes d'inférence OWL
- vérification consistance des ontologies
- Support pour des règles en langage SWRL

Exemple Tom et Jerry

Construire un modèle, une ontologie OWL, des concepts et des propriétés

```
OntModel om = ModelFactory.createOntologyModel();
om.setNsPrefix("animal", animal_ns);
om.setNsPrefix("rdf", RDF.getURI());
om.setNsPrefix("rdfs", RDFS.getURI());
om.setNsPrefix("xsd", XSD.getURI());
OntClass p = om.createClass(animal_ns + "Personnage");
OntClass chat = om.createClass(animal_ns + "Chat");
OntClass souris = om.createClass(animal_ns + "Souris");
chat.addSuperClass(p);
souris.addSuperClass(p);
souris.addDisjointWith(chat);
SymmetricProperty iAvec =
om.createSymmetricProperty(animal_ns + "ineragitAvec");
DatatypeProperty age = om.createDatatypeProperty(animal_ns + "age");
age.addRange(XSD.nonNegativeInteger);
age.convertToInverseFunctionalProperty();
```

Listing 2: premiers concepts

Exemple Tom et Jerry

Ajouter des comportements aux propriétés

```
ObjectProperty chasse =  
    om.createObjectProperty(animal_ns + "chasse");  
ObjectProperty chasse_par =  
    om.createObjectProperty(animal_ns + "chassePar");  
chasse.addInverseOf(chasse_par);  
chasse.setDomain(chat);  
chasse.setRange(souris);
```

Listing 3: ajouts de propriétés

Exemple Tom et Jerry

Définir les concepts au travers de restriction

```
AllValuesFromRestriction onlyPersonnage =  
om.createAllValuesFromRestriction(null, iAvec, p);  
p.addSuperClass(onlyPersonnage);  
  
SomeValuesFromRestriction somePersonnage =  
om.createSomeValuesFromRestriction(null, iAvec, p);  
p.addSuperClass(somePersonnage);
```

Listing 4: classe basée sur une restriction

Exemple Tom et Jerry

Définir des individus

```
Individual tom = chat.createIndividual(animal_ns + "Tom");
tom.addProperty(RDF.type, chat);
tom.addProperty(RDFS.label, "Tom");

Individual jerry = souris.createIndividual(animal_ns + "Jerry");
jerry.addProperty(RDF.type, souris);
jerry.addProperty(RDFS.label, "Jerry");

DatatypeProperty color = om.createDatatypeProperty(animal_ns + "couleur");
tom.addProperty(color, om.createLiteral("gris", "fr"));
jerry.addProperty(color, om.createLiteral("beige", "fr"));

tom.addProperty(age, om.createTypedLiteral("4", XSD.getURI() + "int"));
jerry.addProperty(age, om.createTypedLiteral("2", XSD.getURI() + "int"));

tom.addProperty(iAvec, jerry);
```

Listing 5: ajouts d'individus

Exemple Tom et Jerry

Consistance des énoncés (classifieur interne)

```
String fic = "tomOnt.n3";
OntModel om =
    ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM_RDFS_INF);
try {
    om.read(fic);

    ValidityReport validity = om.validate();
    if (validity.isValid())
        System.out.println("valid");
}
finally {om.close();}
}}
```

Listing 6: validité de l'ontologie

Exemple Tom et Jerry

Consistance des énoncés (classifieur OWL interne)

```
public static void main(String args[]) {
    String fic = "tomOnt.n3";
    OntModel om = ModelFactory.createOntologyModel();
    try {
        om.read(fic);
        Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
        reasoner = reasoner.bindSchema(om);
        InfModel infmodel = ModelFactory.createInfModel(reasoner, om);

        ValidityReport validity = infmodel.validate();

        if(validity.isValid()) System.out.println("OK");
        else{System.out.println("Conflicts");
        for(Iterator i = validity.getReports(); i.hasNext(); )
        {ValidityReport.Report report =(ValidityReport.Report)i.next();
        System.out.println(" "+ report);}}
    }
    finally {om.close();}
}
```

Listing 7: validité de l'ontologie

Avec qui interagit Jerry

```
OntModel om = ModelFactory.createOntologyModel();
om.read("tomOnt.n3");
String animal_ns = om.getNsPrefixURI("animal" );
    OntClass jerry = om.getOntClass(animal_ns + "Jerry");
    OntProperty iAvec = om.getOntProperty(animal_ns + "interagitAvec");

Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
reasoner = reasoner.bindSchema(om);
InfModel infmodel = ModelFactory.createInfModel(reasoner, om);
printStatements(infmodel, jerry, iAvec, null);
}

public static void printStatements(Model m, Resource s, Property p, Resource o) {
    for (StmtIterator i = m.listStatements(s,p,o); i.hasNext(); ) {
        Statement stmt = i.nextStatement();
        System.out.println(" - " + PrintUtil.print(stmt));
    }
}
}
```

Listing 8: inférence