Rapport de stage

Construction d'une ontologie des UCD en OWL avec utilisation de l'éditeur d'ontologie Protégé

Sébastien Derivaux*
en collaboration avec Rim Al Hulou
sous la direction de Sébastien Derrière et Amedeo Napoli

31 août 2004

Table des matières

*esotech@free.fr

Introduction

Mon stage s'est déroulé dans le cadre de l'ACI MDA[1] (Action Concertée Incitative Masses de Données) durant l'été 2004. Les documents relatifs à mon stage se trouvent sur le wiki du projet[2].

Une ontologie est un ensemble de concepts, structurés par des relations entre eux, avec notamment la relation de subsomption (un concept père est plus général que son descendant). Les ontologies peuvent avoir deux objectifs, le premier *descriptif* est de former une base de connaissance servant de norme à un ensemble de clients. Le second de *raisonnement* utilise le fait que cette connaissance est formelle pour faire des inférences. On peut ainsi trouver que deux concepts qui ne sont pas liés explicitement dans l'ontologie sont équivalents.

Les UCD (Unified Content Descriptor) [DGM+04] sont un moyen standardisé de décrire les quantités manipulées en astronomie. Par exemple `phot.mag;em.opt.R` représente une magnitude mesurée dans le rouge. Un UCD est constitué de mots (comme `phot.mag` et `em.opt.R`) pris dans la liste standardisée [DM04] séparés par ';'. Chacun de ces mots est composé d'atomes (comme `phot` et `mag`) séparés par '.' . Le symbole '.' note une spécialisation, ainsi `phot.mag` (magnitude) est un sous concept de `phot` (mesure photométrique).

Le but du stage était de formaliser la connaissance sur les UCD pour en faire une ontologie. Le travail a comporté deux aspects :

- Construction d'une ontologie explicitant les règles syntaxiques de fabrication d'UCD.
- Mise au point d'une ontologie pour la classification des UCD en fonction de leur sémantique à l'aide d'un formulaire.

1 Stockage des règles syntaxiques des UCD dans l'ontologie

Le premier travail a été de fournir un outil permettant de vérifier la validité d'un UCD selon les règles de construction des UCD [DGM+04].

Le programme est décomposé en un serveur java et une interface web en PHP, avec une communication TCP/IP entre les deux parties. Le chargement de l'ontologie au format OWL est réalisé par la bibliothèque OWLapi.

L'ontologie est descriptive avec comme concepts les mots UCD et certains métaconcepts comme `primary` qui est le père de tous les mots pouvant se trouver en position primaire. La figure 1 montre la représentation de la hiérarchie dans Protégé.

Deux types de propriétés ont été utilisées, *need* et *allow* qui ont pour domaine et co-domaine les mots UCD. Ils représentent la règle qu'un mot doit être suivi d'un autre (*need*) et qu'un mot peut être suivi d'un autre (*allow*).

Un problème se pose si un mot a plusieurs règles d'un même type, laquelle est à satisfaire en premier ? Par exemple `pos.eq.ra;stat.max;meta.main` est valide car `pos.eq.ra` est un mot primaire et est en relation de type `allow` avec `stat.max` et `meta.main`. Or `pos.eq.ra;meta.main;stat.max` n'est pas valide

[1] `http://cdsweb.u-strasbg.fr/MDA/mda.html`

[2] `http://cds.u-strasbg.fr/twikiMDA/bin/view/Ontologies/StageOntoUCD2004`

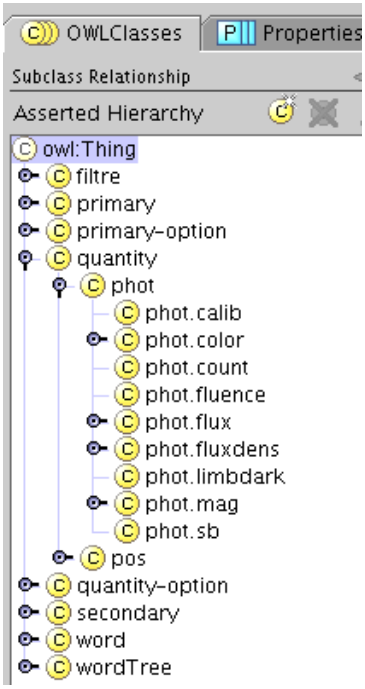

FIG. 1 – Les metaconcepts de l'ontologie avec la branche *quantity* développée

bien qu'il puisse le sembler. La relation `allow stat.max` est *prioritaire* sur celle `allow meta.main`. Il faut donc modéliser une structure de type liste.

1.1 Gestion des listes en OWL

Pour modéliser les listes en OWL, il faut revenir à la définition des listes des langages fonctionnels. Une liste est définie d'une façon récursive : elle est composée d'un premier élément et d'un reste qui est une liste. Par exemple, une liste contenant les concepts `one`, `two` et `three` s'écrit en logique de descriptions :

```
(AND
ilexist first one
ilexist rest (AND
     ilexist first two
     ilexist rest (AND
          ilexist first three
          )
     )
)
```

où "first" est le rôle utilisé pour indiquer le premier élément d'une liste et "rest" est le rôle utilisé pour indiquer son reste.

Même si cette méthode est utilisable, elle est très compliquée à mettre en œuvre. En plus, il n'est pas possible pour l'instant de gèrer ces listes par un raisonneur de logique de descriptions, il n'existe pas de système opérationnel

qui permet de mener de raisonnements sur les listes et de dire, par exemple, que la liste [1,2] est une sous-liste de [1,2,3]. L'idéal serait d'avoir une intégration d'un langage de plus haut niveau pour définir les listes. Cette problématique a été mise de coté.

1.2 Importation de la liste des mots UCD

Le nombre de mots UCD étant important (plus de 500 à ce jour), j'ai développé une application important tous les mots UCD à partir du fichier de référence[DM04]. Grâce aux informations du fichier de référence les mots sont classés sous la branche *primary* s'ils sont primaires et *secondary* s'ils sont secondaires.

On se donne un fichier OWL correspondant à une ontologie de base contenant les concepts *primary*, *secondary*, etc. L'outil permet d'enrichir l'ontologie de base afin d'y inclure tout les mots UCD. Ensuite, l'outil Protégé permet d'y inclure les règles syntaxiques. On obtient donc ainsi un fichier OWL comportant tous les éléments requis pour la validation syntaxique.

L'outil permet aussi de lire directement des règles dans le fichier de référence, cela permet de se passer de l'éditeur Protégé et de centraliser toutes les informations dans un seul fichier.

L'outil ne faisant qu'ajouter des mots UCD à l'ontologie, il est tout à fait possible d'enrichir une ontologie où certains mots sont déjà présents avec des règles associées. Cela peut être utile si le fichier de référence des mots UCD comporte de nouveaux mots.

Le programme a été réalisé en faisant un parseur avec JavaCC (un parseur java) et OWLapi pour la gestion de l'ontologie.

2 Classification par formulaire

La deuxième réalisation a été un système permettant à l'utilisateur de classer des concepts à partir d'un formulaire web que l'utilisateur remplit. Une fois le formulaire rempli, le système se charge d'identifier les concepts équivalents. Si un des concepts subsumant ou équivalent au concept ainsi créé est dans le domaine d'une propriété qui n'est pas dans le formulaire, celle-ci est rajoutée et l'utilisateur a la possibilité de raffiner sa requête.

La mise en oeuvre de l'application s'est faite en java/jsp sous une infrastructure Tomcat. La librairie Jena est utilisé pour la manipulation de l'ontologie au format OWL et l'interaction avec le raisonneur Racer.

2.1 Format de l'ontologie

Le système n'est pas dépendant d'une ontologie particulière ; le système est très général. L'utilisateur peut demander au système de charger une ontologie de son choix (donnée sous la forme d'une URL).

Pour fonctionner avec le système, l'ontologie doit satisfaire certaines contraintes :

- Elle doit posséder l'annotation chaîne de caractères *rfds :Label* dans ses métadonnées. Cette annotation sera utilisée comme titre des pages web
- Certains concepts doivent avoir l'annotation booléenne *isForm* vraie afin d'être considérés comme des formulaires. Les champs des formulaires sont

les propriétés qui ont pour domaine le concept formulaire ou un concept le subsumant (hormis owl :Thing).
- Le nom des concepts doit utiliser les caractères autorisés par OWL, ';' par exemple est interdit, nous avons utilisé '..' pour remplacer les occurences de ';'. Une autre solution serait d'utiliser les champs *rdfs :Label* pour l'affichage, mais cela demande de changer le programme.

2.2 Classification d'UCD

La première mise en œuvre du système permet de retrouver un UCD en remplissant un formulaire. Le système a été testé sur une dizaine d'UCD et fonctionne.

Par exemple, si l'utilisateur prend le formulaire position, le système lui demande si la position utilise le référentiel galactique ou équatorial, si l'utilisateur choisit galactique, l'ontologie trouve l'équivalence du concept produit avec `pos.gal` qui dispose en outre de la propriété coordonnée (latitude ou longitude), l'utilisateur pourra répondre latitude et le système lui dira que son concept se trouve être `pos.gal.lat`.

Néanmoins, il n'est pas facile de définir tous les UCD à partir de concepts simples. Définir une connaissance formelle sur tous les mots UCD serait une tache longue et complexe.

2.3 Classification de sources astronomiques

Une autre application est la classification de sources astronomiques Par exemple, la classification d'étoiles en différentes catégories peut être décrite dans un fichier OWL.

Une fois ce fichier chargé, un formulaire permet de déterminer progressivement à quelle catégorie l'étoile appartient en posant des contraintes sur les paramètres proposés.

La principale limitation pour une utilisation pratique de ce genre d'outil est l'impossibilité d'utiliser des contraintes quantitatives en l'état actuel de OWL (par exemple définir des classes d'objets pour certains intervalles numériques de température ou de luminosité). Seules des contraintes qualitatives peuvent être exprimées, ce qui limite sérieusement les applications scientifiques.

3 Outils

3.1 Le format OWL

OWL[3](Ontology Web Langage) est un langage de type XML permettant de représenter des ontologies. Il est standardisé au niveau de l'organisme W3C. Il comprend trois niveaux de langage ; OWL Lite, OWL DL et OWL Full qui correspondent à une expressivité de plus en plus riche. OWL DL est prévu pour les raisonneurs sur les logiques de descriptions [Nap97], c'est donc celui-ci que nous avons utilisé. OWL DL impose par exemple qu'une classe ne peut pas être vue comme une instance, ce que permet OWL Full.

[3] `http://www.w3.org/2004/OWL/`

3.2 Protégé

Pour la durée du stage nous avons utilisé Protégé 2.1.2[4] avec le plugin OWL 1.2[5]. Protégé est un éditeur d'ontologie et le plugin OWL permet de développer l'ontologie en OWL. L'interface est très intuitive et le logiciel est assez mature.

3.3 DIG

DIG[6] est une spécification d'interface communément utilisé par les raisonneurs comme Racer. Il est basé sur HTTP et XML. Il permet d'exprimer des logiques de description du type $SHOIQD_n^-$.

3.4 Racer

Racer[7] est un raisonneur supportant la logique de description $ALCQHIR+(D)-$ ($SHIQ$ plus le domaine concret). Il est très simple d'utilisation et possède une interface DIG ce qui permet de l'interfacer très facilement. Il gratuit pour la recherche mais n'est malheureusement pas open source.

3.5 Librairies OWL

OWL étant un format assez répandu, un certain nombre de bibliothèques fournissent des méthodes pour interagir avec des ontologies dans ce format. Deux bibliothèques ont été utilisées durant ce stage.

3.5.1 OWLapi

La bibliothèque OWLapi[8] a été la première à avoir été utilisée. Elle est malheureusement trop jeune et très complexe à utiliser. Tout passe par l'usage du pattern du visiteur, ce qui fait qu'il faut écrire une classe pour chaque requête (par exemple avoir la liste des superclasses). Il manque aussi une documentation autre que la javadoc.

3.5.2 Jena

Jena[9] est la bibliothèque utilisée par Protégé. Elle est beaucoup plus facile à utiliser que OWLapi et la version CVS permet d'interagir avec un raisonneur DIG comme RACER. Elle n'est pour autant pas exempte de problèmes comme le fait qu'elle n'accepte pas les ontologies avec des accents et certaines fonctions ne renvoyent pas les résultats escomptés.

Conclusion

Comme nous l'avons vu, le format OWL permet de représenter une ontologie de façon très simple et peut s'utiliser très facilement pour une gamme

[4] `http://protege.stanford.edu/`
[5] `http://protege.stanford.edu/plugins/owl/`
[6] `http://dig.sourceforge.net/`
[7] `http://www.cs.concordia.ca/~haarslev/racer/`
[8] `http://sourceforge.net/projects/owlapi`
[9] `http://jena.sourceforge.net/`

d'applications étendue. Il est facilement éditable avec le logiciel Protégé.

Le format OWL pose néanmoins quelques limitations pour exprimer des contraintes d'ordres et des contraintes quantitatives. Ces contraintes sont pénalisantes dans le domaine de l'astronomie et devront être levées soit au niveau du standard, soit en étendant le langage nous-même (le format étant basé sur XML).

Références

[Bec03] Sean Bechhofer. The dig description logic interface : Dig/1.1. Technical report, 2003. URL : `http://dl-web.man.ac.uk/dig/2003/02/interface.pdf`.

[DGM+04] Sebastien Derriere, Norman Gray, Robert Mann, Andrea Preite Martinez, Jonathan McDowell, Thomas Mc Glynn, Francois Ochsenbein, Pedro Osuna, Guy Rixon, and Roy Williams. Ucd (unified content descriptor) - moving to ucd1+ version 1.05, 2004. proposed recommendation URL : `http://www.ivoa.net/Documents/latest/UCD.html`.

[DM04] S. Derriere and A. Preite Martinez. The ucd1+ controlled vocabulary, 2004. working draft URL : `http://www.ivoa.net/internal/IVOA/IvoaUCD/WD-UCDlist-20040823.pdf`.

[Nap97] Amedeo Napoli. Une introduction aux logiques de descriptions, 1997. rapport INRIA n° 3314.