Construction d'une ontologie de descripteurs UCD en astronomie

Emmanuel Nauer, Alexandre Richard, Sébastien Derriere, Françoise Genova, Amedeo Napoli, Yannick Toussaint

Construction d'une ontologie de descripteurs UCD en astronomie

Emmanuel Nauer[*] — **Alexandre Richard**[*] — **Sébastien Derriere**[**] — **Françoise Genova**[**] — **Amedeo Napoli**[*] — **Yannick Toussaint**[*]

[*] *LORIA, campus scientifique, B.P. 239, 54506 Vandœuvre-les-Nancy CEDEX.*
Email : {nauer,napoli,yannick}@loria.fr
[**] *ULP/CNRS - CDS, 11 rue de l'Université, 67000 Strasbourg.*
Email : {derriere,genova}@astro.u-strasbg.fr

RÉSUMÉ. *Les descripteurs* UCD *(Unified Content Descriptors) permettent de décrire des données d'astronomie. Un processus de standardisation, qui est resté sur un plan purement syntaxique jusqu'à présent, s'est avéré nécessaire et aucune sémantique formelle n'a été associée aux descripteurs normalisés. Dans cet article, nous présentons une méthodologie pour construire une ontologie des* UCD *normalisés en* OWL. *Cette ontologie est ensuite exploitée pour l'attribution semi-automatique d'*UCD *à des descriptions de propriétés d'objets célestes issues de catalogues d'astronomie. Cette procédure exploite une classification approchée et progressive s'appuyant sur des méta-données qui établissent des liens entre les éléments lexicaux employés dans les descriptions et les propriétés des objets représentant les* UCD *dans l'ontologie.*

ABSTRACT. *The Unified Content Descriptors (*UCD*) provides a formal vocabulary for astronomical data. A syntaxic standardisation of* UCD *has been set in order to control the vocabulary used for the description of astronomical measures, but no formal semantics has been defined for these descriptors. In this paper, we present a methodology for building an* OWL *ontology about these normalized* UCD. *This ontology is then exploited to retrieve the* UCD*s representing on the best way a description of an astronomical object given by a set of properties. An approximate 2-step classification is performed exploiting metadata describing links between lexical items used in the descriptions and objects properties used to define the* UCD *in the ontology.*

MOTS-CLÉS : *Construction d'ontologie, classification approchée, classification progressive,* OWL, *logiques de descriptions,* UCD

KEYWORDS: *Ontology building, approximate classification,* OWL, *description logics,* UCD

1. Introduction

Les descripteurs UCD (*Unified Content Descriptors*) permettent de décrire des corps célestes en astronomie. Cependant, aucune sémantique formelle ne leur est initialement associée. Par exemple, `pos.eq` est le descripteur UCD qui doit être utilisé pour décrire une position en coordonnées équatoriales, sans définir ce qu'est une position en coordonnées équatoriales. Notre objectif ici est de construire une ontologie, nommée $\mathcal{O}_{\mathcal{UCD}}$, des descripteurs UCD. La construction d'une telle ontologie a pour but d'attribuer une définition et une sémantique aux UCD. Ce travail a nécessité de définir, en collaboration avec un expert du domaine, chacun des descripteurs UCD par un ensemble de *propriétés*. Une des utilisations de l'ontologie $\mathcal{O}_{\mathcal{UCD}}$ concerne l'annotation automatique de table de données d'astronomie.

Dans la suite, nous nous intéressons à trois types de données : (1) des tables de données `objets x propriétés` où les objets sont principalement des corps célestes et les propriétés des mesures ou des informations sur ces mesures, (2) des méta-données qui décrivent les propriétés des tables de données, et (3) des UCD représentant de façon standardisée ces propriétés. L'objectif de l'annotation des tables est d'identifier dans l'ontologie $\mathcal{O}_{\mathcal{UCD}}$ le concept — représentant un descripteur UCD— qui correspond au mieux à chacune des propriétés de la table. La méthode mise en œuvre pour l'annotation des propriétés consiste alors à définir un objet à classer en fonction d'un ensemble de propriétés, puis de rechercher par une classification approchée le ou les concept(s) qui partagent un maximum de propriétés avec l'objet à classer.

En outre, cette expérience de conception a permis d'établir un ensemble de règles générales pour la construction d'ontologies ainsi que pour l'annotation par l'intermédiaire d'un processus de classification.

Cet article se décompose en quatre parties. Nous exposons tout d'abord la problématique générale de notre travail, en parallèle avec une présentation des données d'astronomie. Nous exposons ensuite comment nous avons construit l'ontologie $\mathcal{O}_{\mathcal{UCD}}$ en OWL. Nous détaillons enfin comment nous mettons en œuvre une classification approchée et progressive de concepts dans $\mathcal{O}_{\mathcal{UCD}}$ et discutons nos résultats. Nous concluons par les perspectives de notre travail.

2. Problématique

Ce travail s'inscrit dans le cadre de l'ACI *"Masse de données en Astronomie"* qui se fait en collaboration avec les chercheurs du CDS, Centre de Données en astronomie de Strasbourg[1]. Ce projet inclut des travaux sur les descripteurs UCD qui sont appelés à devenir les éléments du vocabulaire standardisé et universel de description des corps célestes en astronomie, et dont l'utilisation a pour but d'offrir une plus grande interopérabilité entre les sources de données astronomiques. Parmi ces sources, nous nous intéressons plus particulièrement dans notre travail à l'exploitation de catalogues d'astronomie pour lesquels il existe des propositions de standards de

1. http ://cdsweb.u-strasbg.fr/

description [OCH 00]. Un catalogue contient en particulier des tables de données astronomiques et un fichier nommé `ReadMe` qui fournit des méta-données relatives aux tables de données, permettant ainsi l'interprétation de ces tables. La figure 1 illustre ces deux types de données. Chaque colonne de la table de données correspond à un type de mesure ou d'observation qui est décrite par un `Label` composé d'un seul mot (exemple : `RAh`). Ce label est explicité dans le fichier `ReadMe` par un élément `Units` qui indique l'unité de mesure (dans le cas où la colonne décrite contient des mesures) et par un élément `Explanations` qui donne une courte explication textuelle associée au label de la colonne.

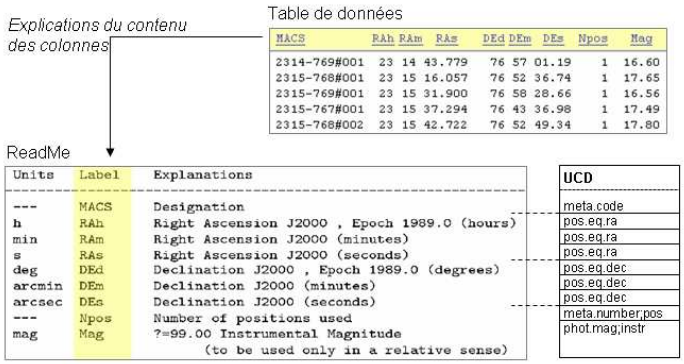

Figure 1. *Exemple d'une table de données, d'un fichier `ReadMe` et des* UCD *associés. Chaque ligne de la table de données correspond à la description d'un objet céleste, chaque colonne représente une propriété de cet objet. Le contenu de chaque colonne de la table est explicité par une ligne dans le fichier `ReadMe` associé.*

Le problème d'association d'un descripteur UCD à une colonne de la table de données est traité à partir du contenu des trois colonnes du fichier `ReadMe`, c'est-à-dire `Units`, `Label` et `Explanations` ; associer un UCD à une colonne de la table de données revient à associer un UCD à la ligne du `ReadMe` qui décrit la colonne. Le résultat escompté est donné en illustration sur la figure 1 par les UCD correspondant effectivement à chacune des lignes du `ReadMe`. Par exemple, `pos.eq.ra` est le descripteur UCD représentant une ascension droite (*right ascension*) ; il doit être associé aux colonnes `RAh`, `RAm` et `RAs` de la table de données. En réalité, cela correspond à la mesure d'une ascension droite en heures, minutes et secondes.

2.1. *Présentation des* UCD

Les UCD sont des descripteurs formels de données astronomiques sur les corps célestes [DER 04a] contrôlés par l'Alliance de l'Observatoire Virtuel International (IVOA[2]). La création des UCD répond à plusieurs besoins : (1) éviter les ambiguïtés

2. http ://www.ivoa.net/

(un descripteur pour un type de données précis) et (2) éviter les redondances (un descripteur et un seul pour un même type de données). Les UCD servent à décrire des mesures ou des informations sur ces mesures, notamment le type des mesures. Par exemple, `phys.temperature` est un UCD qui renvoie à une température. Les UCD ont été créés il y a une dizaine d'années et ont déjà connu une révision avec le passage de la version UCD1 à la version UCD1+ [DER 04a] [DER 04b]. Cette révision s'est en particulier traduite par l'adoption d'une nouvelle syntaxe. Nos travaux concernent les UCD1+ et toutes les références aux UCD dans ce papier renvoient aux UCD1+.

L'objectif principal des UCD est de permettre l'interopérabilité entre des sources de données hétérogènes (provenant d'observatoires différents), palliant notamment des hétérogénéités telles que des descriptions dans des langues différentes (par exemple : *"Motion in Right Ascension"* et *"Mouvement mesuré en ascension droite"* font, tous les deux, référence au concept du mouvement d'un corps céleste décrit par l'évolution de son ascension droite) ou des raccourcis d'écriture différents pour une même description (par exemple : *posang*, *pa* et *apa* sont trois raccourcis différents de l'expression *"position angle"* qui exprime une orientation décrite par un angle).

2.2. *Syntaxe des* UCD

L'ensemble des règles assurant la validité d'un UCD est détaillé dans [DER 04a] et [DER 04b] et une synthèse peut être consultée dans [RIC 05]. Nous nous limitons ici aux points essentiels à la compréhension de notre travail.

Définition (donnée dans [DER 04a]) : *un* UCD *est une chaîne de caractères constituée de mots, eux-mêmes constitués d'atomes. Les mots sont séparés par des points-virgules et les atomes par des points*.

Ainsi, par exemple :

- `pos`, `eq`, `ra`, `meta` et `main` sont des *atomes*
- `pos.eq.ra` et `meta.main` sont des *mots*
- `pos.eq.ra;meta.main` est un UCD

Un UCD est composé au minimum d'un mot, et chaque mot est composé d'au moins un atome. Le contrôle des UCD vient du fait que les mots sont validés par un comité scientifique mis en place par l'IVOA. Par conséquent, les briques de base des UCD sont les mots autorisés et non les atomes. Les mots sont regroupés dans une hiérarchie (en fonction des atomes par lequels ils commencent) comprenant 12 catégories pincipales. Par exemple, les mots commençant par `pos` constituent la catégorie des positions.

Combinaison de mots pour la construction d'UCD composés : chaque mot valide est associé à un code qui indique quel rôle il peut jouer dans un UCD composé :

- code P : mot se trouvant nécessairement en première place dans un UCD
- code S : mot ne pouvant être en première place dans un UCD
- code Q : mot pouvant être à une place quelconque dans un UCD

– etc.

Pour être valide, un UCD doit être composé uniquement de mots valides ; les mots qui le composent doivent tous respecter la règle correspondant au code qui leur est associé. Une grammaire est disponible dans [RIC 05]. Dans la suite, nous utilisons *"*UCD *simple"* pour désigner un UCD formé d'un seul mot et *"*UCD *composé"* pour un UCD formé de plusieurs mots.

Interprétation : une interprétation concrète et non formelle en langage naturel est asscociée à chaque UCD. Par exemple, `pos` représente le concept de position, `stat.error;pos.eq.ra` représente une erreur sur la mesure d'une ascension droite, etc.

2.3. *Pourquoi une ontologie des* UCD *?*

Une ontologie permet de représenter formellement un ensemble de concepts en leur attribuant une sémantique. Les avantages qui découlent de l'existence d'une ontologie peuvent être classés en deux grandes catégories [STA 04] [USC 96]. D'un point de vue *"communication et interopérabilité"*, une ontologie facilite l'échange d'information ou de connaissances venant de sources hétérogènes entre des êtres humains et/ou des machines, car elle permet la désambiguïsation et la vérification de cohérence des éléments représentés. D'un point de vue *"spécification, intégration et réutilisation"*, une ontologie peut aider à la spécification de problèmes liés au domaine qu'elle représente et servir de passerelle pour faciliter l'intégration d'une application à une plate-forme. Enfin, même si la construction de l'ontologie est influencée par les applications, tant que la structure de l'ontologie n'est pas remise en cause, l'ontologie et les applications peuvent évoluer séparément. Concrètement, disposer d'une ontologie permet de résoudre certains problèmes comme guider l'utilisateur lors d'un accès à des données [AQU 05] [SAF 04] ou encore organiser et/ou comparer des documents par rapport à leur contenu [HUL 04].

$\mathcal{O}_{\mathcal{UCD}}$ permet, d'un point de vue représentation, de garantir la cohérence et la non-ambiguïté des UCD, ainsi que de les organiser/classifier les uns par rapports aux autres. Elle est utilisée pour la recherche d'information et pour détecter des erreurs ou omissions dans des descriptions, par exemple, pour vérifier la correction de lignes de `ReadMe`. Pour illustration, dans la ligne de `ReadMe` suivante :

```
Units  Label  Explanations
 ---   RAhms  Right ascension in h m s, ICRS(J1991.25)(T3)
```

les unités ont été mentionnées dans la colonne `Label`. Or, $\mathcal{O}_{\mathcal{UCD}}$ spécifie que le concept définissant l'UCD *ascension droite* de cette ligne de `ReadMe` possède une propriété `hasAngleUnit`, ce qui signifie que la colonne `Units` doit être renseignée. Ceci permet donc de détecter une omission sur cette ligne et, éventuellement, de la corriger.

3. Construction d'une ontologie des UCD en OWL

Beaucoup de propositions ont été faites pour définir ce qu'est une ontologie. La plus connue en informatique est probablement celle de Gruber [GRU 93], à savoir

qu'*une ontologie est la spécification explicite d'une conceptualisation*. Dans notre acception de cette définition, le terme *conceptualisation* renvoie à un modèle abstrait prenant la forme d'un ensemble de définitions de concepts et de propriétés de concepts organisés en hiérarchie. Par ailleurs, l'expression *spécification explicite* renvoie au fait que le modèle doit être représenté dans un langage de représentation des connaissances (muni d'une syntaxe et d'une sémantique associée) pour que le modèle soit utilisable aussi bien par des machines que des êtres humains [FEN 03].

3.1. *Formalisme et composants des ontologies*

$\mathcal{O}_{\mathcal{UCD}}$ est codée en OWL-DL qui repose sur l'utilisation du formalisme de représentation des logiques de descriptions [BAA 03]. Ce formalisme s'appuie sur trois types d'entités :

– les *concepts* (ou *classes*) qui représentent des ensembles d'individus ayant des propriétés communes. L'ensemble des individus correspond à l'*extension* du concept. Par exemple, le concept `Measure` représente un ensemble de mesures ; cet ensemble de mesures est l'extension du concept `Measure`.

– des *individus* (ou *instances* des concepts). Par exemple, les individus `100 mètres` et `45 degrés` sont des instances du concept `Measure`.

– les *rôles* (ou *propriétés*) qui représentent des relations binaires entre les concepts. Par exemple, le rôle `hasUnit` entre le concept `Measure` et le concept `Unit` exprime qu'une mesure (instance du concept `Measure`) est en relation avec une unité (instance du concept `Unit`). Chaque rôle possède un *domaine* et un *co-domaine* : le domaine est le concept où est déclaré le rôle et le co-domaine est le concept avec lequel le rôle établit une relation. Ainsi, `hasUnit` a pour domaine le concept `Measure` et pour co-domaine le concept `Unit`.

Les concepts (et les rôles le cas échéant) sont organisés en une hiérarchie par la relation de *subsomption*, notée $\sqsubseteq$, où $D \sqsubseteq C$ se lit "C subsume D" ou "D est subsumé par C". Un concept C subsume un concept D si et seulement si C est plus général que D. On note $\top$ le concept le plus général, qui est à la racine de la hiérarchie de concepts et subsume tous les autres concepts. .

Un concept définit des conditions d'appartenance d'un individu à l'extension de ce concept. Suivant ces conditions, le concept est *primitif* ou *défini* :

– s'il ne s'agit que de conditions nécessaires, alors le concept est *primitif*. Par exemple, le concept `Unit` introduit par `Unit` $\sqsubseteq \top$ est primitif.

– s'il s'agit d'un ensemble de conditions nécessaires et suffisantes (une instance X fait partie de l'extension d'un concept C si et seulement si X a les mêmes propriétés que C), alors le concept est *défini* et l'ensemble de conditions constitue la *définition* de ce concept. Les concepts définis sont introduits par une équivalence notée $\equiv$, où $C \equiv D$ signifie $C \sqsubseteq D$ et $D \sqsubseteq C$. Par exemple, le concept `Measure` introduit par `Measure` $\equiv$ (`hasUnit` = 1) $\sqcap$ (`hasOneValue` = 1) est défini. Si `X` a une valeur et une unité alors `X` est une mesure ; à l'inverse, une mesure possède exactement une unité et

une valeur.

Les définitions de concepts font essentiellement intervenir les constructeurs suivants :

– la *conjonction de concepts*, notée $\sqcap$,

– la *cardinalité* qui fixe le nombre minimal et maximal de valeurs élémentaires que peut prendre un rôle. Elle est notée $\leq$, $\geq$ ou = suivant qu'il s'agit d'une cardinalité maximale, minimale ou exacte.

$\mathcal{O}_{\mathcal{UCD}}$ a été codée en OWL par l'intermédiaire de l'éditeur d'ontologies PROTÉGÉ [3] [HOR 04]. Le système RACER a été utilisé comme moteur de subsomption et de classification [HAA 01].

3.2. *Construction d'une ontologie des mots* `pos`

Il n'existe pas de méthode unifiée de création d'une ontologie, mais le processus itératif présenté sur la figure 2, introduit dans [STA 04] [USC 95], donne les grandes lignes de la conception d'une ontologie.

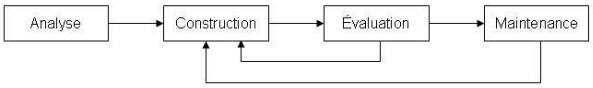

Figure 2. *Phases de construction d'une ontologie.*

La phase d'analyse nécessite de déterminer ce que conceptualise l'ontologie et quelles sont les utilisations qui vont en être faites. La phase de construction s'attache à la mise en place des hiérarchies de concepts et de rôles ainsi qu'aux définitions de concepts. La phase d'évaluation a pour but de tester la cohérence de l'ontologie et d'effectuer des tests d'utilisation dans le but de corriger ou d'ajuster les éléments nécessaires (par un retour à la phase de construction). La phase de maintenance concerne l'utilisation effective de l'ontologie ; un retour à la phase de construction permet de prendre en compte les évolutions nécessaires.

Lors de la création d'une ontologie, un soin particulier doit être apporté au respect des critères suivants [GRU 93] :

Adéquation : une ontologie doit être adaptée à l'usage qui en est fait. Dans une ontologie trop générale ou trop spécifique, les définitions sont trop vagues ou trop complexes (car mettant en jeu des paramètres n'ayant aucun rapport avec le cadre d'exploitation de l'ontologie).

Clarté : les concepts définis sont toujours préférables aux concepts primitifs.

Cohérence : l'ontologie ne doit pas être incohérente, c'est-à-dire qu'on ne doit pas avoir à la fois les concepts A et $\neg$A.

3. http ://protege.stanford.edu

Extensibilité : une ontologie doit pouvoir être étendue sans remettre en cause sa cohérence. Ceci implique en particulier d'avoir des règles de construction précises.

La construction de l'ontologie nécessite une connaissance du domaine d'application et donc l'intervention d'experts du domaine tant pour la construction elle-même que pour la vérification et la validation de l'ontologie (seule la cohérence peut être vérifiée par un non-expert).

Afin de restreindre la complexité du problème de la construction d'une ontologie des UCD, nous avons limité notre domaine d'étude aux UCD représentant des positions. Les UCD décrivant des positions font principalement intervenir des mots commençant par l'atome `pos`. Ces mots sont représentés par l'expression "mot `pos`" dans ce qui suit. Parmi les mots `pos`, on trouve par exemple :

- `pos` : qui représente le concept de position
- `pos.eq` : qui représente des coordonnées équatoriales
- `pos.eq.ra` : qui représente une ascension droite (en coordonnées équatoriales)

En réponse aux deux grandes questions de la phase d'analyse, il faut construire une ontologie des UCD incluant au moins un mot `pos`, notée $\mathcal{O}_{\mathcal{POS}}$, et pouvoir exploiter $\mathcal{O}_{\mathcal{POS}}$ pour mettre en correspondance une ligne de `ReadMe` avec un UCD. Cependant, la construction de $\mathcal{O}_{\mathcal{POS}}$ se heurte à deux problèmes majeurs. Primo, il n'existe pas de liste exhaustive des UCD incluant des mots `pos`. En effet, seuls les UCD simples sont répertoriés de façon exhaustive, les UCD composés n'étant controlés qu'à travers les règles de composition énoncées précédemment. Secondo, le nombre d'UCD incluant des mots `pos` construits à partir de ces mêmes règles dépasse potentiellement 10^6. Au delà des problèmes de faisabilité, une telle taille rendrait l'ontologie difficilement lisible et utilisable. Mais s'il n'est pas possible de construire une ontologie de tous les UCD incluant un mot `pos`, il est possible de le faire sur la liste des mots `pos` qui contient actuellement 58 mots `pos`. Le nombre réduit de mots rend possible la création manuelle d'une ontologie relative aux mots `pos`. Nous avons donc décidé de construire une ontologie de concepts définissant ces mots et d'adapter la phase d'exploitation de l'ontologie pour pouvoir prendre en compte les UCD composés.

3.3. *Processus de construction de l'ontologie*

Nous avons identifié 5 grandes étapes dans la création de $\mathcal{O}_{\mathcal{POS}}$; nous les détaillons maintenant :

1) **Identification des concepts**
Règle : un concept représentant un *mot* est nécessairement défini et ne peut représenter qu'un seul mot. Pour nous qui cherchons à représenter les mots `pos`, de tels mots sont par exemple : `pos.eq`, `pos.eq.ra`, `pos.pm.ra`. Le nom choisi pour identifier chaque concept est le mot qu'il représente.

2) **Écriture des définitions de concepts et identification des rôles**
Les définitions se ramenant à des ensembles de rôles, établir les définitions des concepts passe par l'identification des rôles que possède un concept. Par exemple :

pour établir la définition de `pos.eq.ra`, nous devons savoir ce qui définit une ascension droite dans notre contexte. Nous obtenons :
- *`hasOneValue = 1` (une ascension droite est une mesure, elle a donc une valeur),*
- *`hasAngleUnit = 1` (cette valeur est exprimée avec une unité d'angle),*
- *`hasFrameTypeEq = 1` (cette mesure est faite dans un repère équatorial),*
- *`hasCcOriginEqRa = 1` (l'ascension droite est mesurée à partir de l'origine des ascensions droites du repère équatorial).*

Écrire ces définitions suppose que nous incluons dans $\mathcal{O}_{\mathcal{POS}}$ les rôles apparaissant dans ces définitions. De même, introduire les rôles implique d'inclure les concepts co-domaines de ces rôles s'ils ne sont pas déjà présents dans $\mathcal{O}_{\mathcal{POS}}$. Pour l'exploitation de $\mathcal{O}_{\mathcal{POS}}$, nous avons souhaité pouvoir identifier un rôle de manière non ambiguë à partir de son co-domaine. Pour permettre cela, nous avons imposé la règle suivante : *un co-domaine unique et différent pour chaque rôle*, de façon à faire correspondre un rôle avec un co-domaine (et inversement). Par exemple, le co-domaine `AngleUnit` ne peut être associé à une définition de concept que par le rôle `hasAngleUnit`. Ce choix, peu contraignant lors de la construction, facilite grandement l'exploitation de l'ontologie car il permet de retrouver les rôles à partir des co-domaines.

3) **Hiérarchisation des concepts**
La hiérarchie des concepts est organisée par la relation de subsomption. Pour les concepts définis, la hiérarchie découle des définitions : si la définition d'un concept `C1` est plus spécialisée que la définition d'un concept `C2`, alors `C1` $\sqsubseteq$ `C2`. Ainsi `pos.barycenter` $\sqsubseteq$ `pos` car `pos` $\equiv$ (`hasFrameType` = 1) et `pos.barycenter` $\equiv$ (`hasFrameType` = 1) $\sqcap$ (`hasBarycentricCoordinates` $\geq$ 2).

Pour les concepts primitifs, la hiérarchisation est donnée par l'expert qui aide à construire l'ontologie du domaine. Ainsi, l'expert a établi pour les deux concepts primitifs `Unit` et `AngleUnit`, la relation suivante : `AngleUnit` $\sqsubseteq$ `Unit`.

4) **Hiérarchisation des rôles**
La hiérarchie des rôles est également organisée par la relation de subsomption, selon la règle suivante : r1 $\sqsubseteq$ r2 *si et seulement si* domaine(r1) $\sqsubseteq$ domaine(r2) et co-domaine(r1) $\sqsubseteq$ co-domaine(r2). Par exemple :
- `hasUnit` a pour domaine `Measure` et pour co-domaine `Unit`
- `hasAngleUnit` a pour domaine `pos.angDistance` et pour co-domaine `AngleUnit`
- `hasAngleUnit` $\sqsubseteq$ `hasUnit` si et seulement si `pos.angDistance` $\sqsubseteq$ `Measure` et `AngleUnit` $\sqsubseteq$ `Unit`.

5) **Amélioration éventuelle de la lisibilité**.
Enfin, on peut souhaiter améliorer la lisibilité de l'ontologie en créant des concepts abstraits, subsumants d'un ensemble de concepts plus spécifiques, à la façon des classes abstraites en programmation par objets. Toujours dans but de lisibilité, nous avons souhaité que ces concepts abstraits respectent la condition suivante : l'expert doit juger que le concept abstrait a un sens par rapport à l'ontologie.

Par exemple : `Measure` $\equiv$ `(hasOneValue = 1)` $\sqcap$ `(hasUnit = 1)` subsume tous les concepts renvoyant à des mesures (les concepts des mesures spécifiques ont tous une valeur et une unité) et représente la mesure en général (ce qui a un sens en astro-

nomie). Il peut donc être intégré à $\mathcal{O}_{\mathcal{POS}}$.

Après avoir décrit notre approche pour la conception de $\mathcal{O}_{\mathcal{POS}}$, nous montrons dans la partie suivante comment nous avons exploité cette ontologie pour associer un UCD à une ligne de `ReadMe`.

4. Attribution d'un UCD à une ligne de `ReadMe`

L'objectif est ici d'attribuer un UCD à une ligne d'un fichier `ReadMe`, à partir du contenu des colonnes `Units`, `Label` et `Explanations`. À l'heure actuelle, le système développé n'associe que des UCD simples ; nous discutons de l'association d'UCD composés en paragraphe 5.

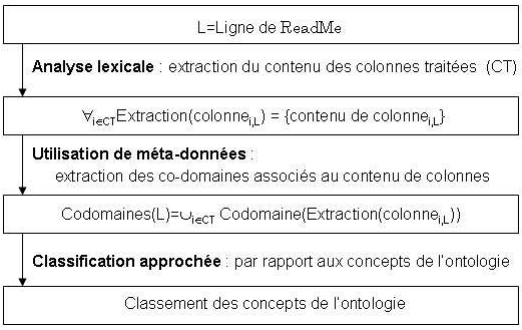

Figure 3. *Schéma global de fonctionnement du système d'attribution d'*UCD.

Le fonctionnement global du système est présenté en figure 3. Nous partons d'une ligne de `ReadMe` pour arriver à une liste triée de concepts pertinents de $\mathcal{O}_{\mathcal{POS}}$. Dans cette liste, les concepts les mieux classés sont ceux qui représentent les mots `pos` dont la description correspond le mieux à la ligne de `ReadMe`. Dans un premier temps, nous n'exploitons que les colonnes `Units` et `Explanations` ; `CT`, qui représente l'ensemble des colonnes traitées, vaut `{Units,Explanations}`. Nous décrivons à présent en détail le fonctionnement du système avec l'exemple de la ligne `L` suivante :

```
Units   Label   Explanations
s       RAs     Right Ascension J2000 (seconds)
```

1) **Analyse lexicale**

La première étape consiste en une analyse lexicale sur la ligne du `ReadMe` pour en extraire le contenu des colonnes `Units` et `Explanations`. En pratique, à partir de la ligne `L` précédente, le système renvoie :

`Extraction(Units,L)={s}`, et

`Extraction(Explanations,L)={right,ascension,J2000,seconds}`.

2) **Utilisation de méta-données établies par un expert du domaine**

Le but de cette seconde étape est d'obtenir un ensemble de co-domaines de rôles à partir des ensembles de termes obtenus à l'étape 1. C'est ici qu'est exploitée l'hypothèse que le co-domaine d'un rôle identifie le rôle de manière unique (hypothèse découlant

de la règle suivie lors de la construction de l'ontologie précisée au paragraphe 3.3 : chaque rôle doit avoir un co-domaine différent et unique) ainsi que des fichiers de méta-données construits avec l'expert pour chaque colonne exploitée, ces fichiers de méta-données ayant pour but à chaque élément lexical de la colonne un ensemble de co-domaines auxquels l'élément lexical fait référence.

En pratique, nous fournissons en entrée les ensembles `{s}` et `{right, ascension,J2000,seconds}`, obtenus à l'étape 1. Les fichiers de méta-données, de la forme : `{(contenu de la colonne => {co-domaines})}` et construits a priori par l'expert permettent d'associer des co-domaines aux entrées lexicales présentes dans les colonnes :

`Méta(Units)={(s=>{AngleUnit,Value},(mas/yr=>{AngularSpeedUnit,Value),...}`
`Méta(Explanations)={(right=>{CcOriginEqRa,Value},(ascension=>{Value}),...}`

Le système donne en sortie l'ensemble des co-domaines associés à la ligne de `ReadMe` traitée, à savoir (en ignorant les doublons) : `Codomaines(L)={AngleUnit,CcOriginEqRa,Value}`.

3) **Classification approchée et progressive**

Dans cette dernière étape, on recherche les concepts de $\mathcal{O}_{\mathcal{POS}}$ possédant les rôles de `Codomaines(L)` (puisque par hypothèse, avoir les co-domaines c'est avoir les rôles) via un parcours de $\mathcal{O}_{\mathcal{POS}}$, puis on classe ces concepts en fonction d'un score égal au nombre de rôles qu'ils partagent avec `Codomaines(L)`. Dans notre exemple, l'ensemble des rôles de `Codomaines(L)` est : `{hasAngleUnit, hasCcOriginEqRa, hasOneValue}`. Le classement des concepts de $\mathcal{O}_{\mathcal{POS}}$ qui partagent des rôles avec `Codomaines(L)` est le suivant (les rôles partagés ont été écrits en italiques) :

CONCEPT	SCORE	ROLES
pos.eq.ra	3	*hasAngleUnit*, *hasOneValue*, *hasCcOriginEqRa*, hasFrameTypeEq
pos.angDistance	2	*hasAngleUnit*, *hasOneValue*
pos.az.alt	2	*hasAngleUnit*, *hasOneValue*, hasCcOriginAzAlt, hasFrameTypeAz
pos.eq.dec	2	*hasAngleUnit*, *hasOneValue*, hasCcOriginEqDec, hasFrameTypeEq
...	...	...

Dans cet exemple, comme il existe un unique meilleur classé, c'est lui que le système désigne comme l'UCD à associer à la ligne de `ReadMe`, à savoir `pos.eq.ra`.

En cas d'égalité au premier rang du classement, le processus de classification est repris, dans un second temps — d'où l'appellation de classification *progressive* — avec `CT={Units,Explanations,Label}` pour exploiter la colonne `Label` du `ReadMe`. Les labels sont, en effet, porteurs d'information importante, mais leur exploitation ne peut avoir lieu que dans une seconde phase car ils sont souvent ambigus. Par exemple, le même label `alpha` peut désigner à la fois (dans deux lignes de `ReadMe`) un angle de phase d'une orbite d'un corps céleste et une ascension droite d'un corps céleste. Il est donc difficile pour l'expert d'associer des co-domaines à un label de manière sûre lors de la construction du fichier de méta-données `Méta(Label)`. De fait, une recherche de concept effectuée avec l'exploitation de la colonne `Label` peut conduire à un résultat erroné ; c'est pourquoi nous n'exploitons cette colonne qu'en cas de nécessité, c'est-à-dire si la première passe débouche sur une égalité au premier

rang.

Enfin, s'il y a toujours égalité au premier rang après la seconde passe, c'est à l'expert de choisir l'UCD correspondant au mieux à une ligne de `ReadMe` parmi les concepts à égalité au premier rang.

4.1. *Évaluation de la méthode*

Pour nos tests, nous avons utilisé des `ReadMe` des catalogues d'astronomie les plus employés et nous avons confronté nos résultats à des attributions d'UCD faites par un expert humain. Pour 75 lignes de `ReadMe` traitées, le bilan est le suivant :

– 4 lignes ont eu un UCD attribué dès la première passe,

– 42 lignes supplémentaires ont eu un UCD attribué après la deuxième passe,

– 26 lignes relèvent du choix d'un expert (égalité au premier rang du tableau) après la deuxième passe,

– 3 lignes ont conduit à un échec (l'UCD attendu n'apparaît pas au premier rang du tableau de concepts renvoyé en fin de traitement).

Potentiellement, une ligne de `ReadMe` fait émerger un certain nombre de co-domaines possibles, ce qui à son tour conduit à un nombre élevé de concepts possibles (partageant les rôles associés à ces co-domaines) comme UCD. Les trois échecs sont dus plus particulièrement à des erreurs d'attribution entre les éléments de la ligne de `ReadMe` et les co-domaines possibles.

4.2. *Comparaison à l'existant*

La seule approche mise en oeuvre jusqu'alors au Centre de Données Astronomiques de Strasbourg pour l'association d'UCD à des lignes de `ReadMe` réside dans une application spécifique nommée *UCD Builder*[4]. Cette application utilise un ensemble de règles lexicales et syntaxiques qui associent chacune des scores à des UCD potentiels ; une liste triée d'UCD est proposée en sortie en fonction d'un score global. Cette application, qui n'exploite que la partie `Explanations`, donne de bons résultats, aussi bien sur des UCD composés que des UCD simples. Par contre, une limitation majeure d'un tel système est la dépendance de ses règles lexicales et syntaxiques aux évolutions des UCD et aux langues naturelles utilisées dans les descriptions textuelles : les règles doivent être réécrites pour tenir compte de chaque changement.

L'approche que nous proposons rend le processus d'identification indépendant de la langue utilisée dans les `ReadMe` et des évolutions des UCD. En effet, dans le premier cas, il suffit de rajouter des entrées lexicales dans les fichiers de méta-données et dans le deuxième cas de s'assurer que l'ontologie est toujours adaptée et de la faire évoluer le cas échéant, sans avoir à changer la procédure d'association d'un concept à une ligne de `ReadMe`. Par conséquent, notre approche a pour avantage d'être moins sensible aux

4. http ://vizier.u-strasbg.fr/UCD/cgi-bin/descr2ucd

évolutions que la méthode employée dans *UCD Builder*. Il reste encore à prendre en compte les UCD composés avant de pouvoir faire une comparaison complète entre les performances de notre système et celles d'*UCD Builder*.

5. Conclusion

Notre stratégie de traitement du cas des UCD simples donne des résultats encourageants. Nous travaillons actuellement à la prise en compte des UCD composés. L'objectif est ici d'exploiter les règles de compositions. Ainsi, à partir des mots obtenus par la procédure d'attribution d'UCD simples et à partir des règles de composition précisant les places potentielles que peuvent occuper les mots dans un UCD composé se calcule un score global pour chaque UCD valide. Par exemple, pour la ligne de `ReadMe` suivante correspondant à l'UCD composé `pos.pm;pos.eq.ra` :

```
Units   Label   Explanations
mas/yr  pmRA    [-4418.0,6544.2]? prop.mot. in RA*cos(dec)
```

la procédure d'attribution UCD simple produit le résultat suivant (`pos.eq.ra` étant mieux classé que `pos.eq.dec` par l'exploitation de la colonne `Label` :

CONCEPT	SCORE	RÔLES PARTAGÉS
pos.eq.ra	3	*hasOneValue*, *hasAngleUnit*, *hasCoOriginEqRa*, hasFrameTypeEq
pos.eq.dec	3	*hasOneValue*, *hasAngleUnit*, *hasCoOriginEqDec*, hasFrameTypeEq
pos.pm	2	*hasOneValue*, *hasAngularSpeedUnit*, hasMotionTypePm
pos.angDistance	2	*hasOneValue*, *hasAngleUnit*
...	...	

L'utilisation des règles de compositions indiquent que les quatre mots présentés ci-dessus possèdent un code `Q` et peuvent donc tous apparaître en première ou seconde place dans une composition. Ainsi tous les couples composés de deux de ces termes, à savoir `pos.eq.ra;pos.eq.dec`, `pos.eq.dec;pos.eq.ra`, `pos.eq.dec;pos.pm`, `pos.pm;pos.eq.ra`, etc., sont valides. L'idée dans le calcul du score global est de trouver le couplage qui maximise le nombre de rôles différents recherchés. Ainsi, on peut noter que seul `pos.pm` possède le rôle *hasAngularSpeedUnit*. Le poids de ce mot dans la recherche d'un UCD composé est donc à majorer. Une telle recherche peut être mise en lien avec la recherche de classes polythétiques, classes composées en partie de disjonction d'attributs introduit par le type UNION dans [CHR 94] [BER 99].

Pour finir, il faut souligner qu'un point à résoudre est de pouvoir déterminer, pour une ligne de `ReadMe` donnée, si la recherche d'UCD concerne un UCD simple ou un UCD composé.

6. Bibliographie

[AQU 05] D'AQUIN M., BOUTHIER C., BRACHAIS S., LIEBER J., NAPOLI A., « Knowledge Editing and Maintenance Tools for a Semantic Portal in Oncology », *International Journal of Computer Human Studies*, vol. 62, n° 5, 2005, p. 619–638.

[BAA 03] BAADER F., CALVANESE D., MCGUINNESS D., NARDI D., PATEL-SCHNEIDER P., Eds., *The Description Logic Handbook*, Cambridge University Press, Cambridge, UK, 2003.

[BER 99] BERTINO E., GUERRINI G., MERLO I., MESITI M., « An Approach to Classify Semi-Structured Objects », GUERRAOUI R., Ed., *Proceedings of ECOOP'99, Lisboa (Portugal)*, vol. 1628 de *Lecture Notes in Computer Science*, p. 416–440, Springer, 1999.

[CHR 94] CHRISTOPHIDES V., ABITEBOUL S., CLUET S., SCHOLL M., « From Structured Documents to Novel Query Facilities », *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1994.

[DER 04a] DERRIERE S., GRAY N., MANN R., MARTINEZ A. P., MCDOWELL J., MCGLYNN T., OSCHENBEIN F., OSUNA P., RIXON G., WILLIAMS R., « UCD (Unified Content Descriptor) - moving to UCD1+. Version 1.06, IVOA Proposed Recommendation 2004-10-26 », http ://www.ivoa.net/documents/latest/ucd.html/, 2004.

[DER 04b] DERRIERE S., MARTINEZ A. P., « The UCD1+ controlled vocabulary, Version 1.06, IVOA Working Draft 2004-08-2 », http ://www.ivoa.net/documents/latest/ucdlist.html/, 2004.

[FEN 03] FENSEL D., HENDLER J., LIEBERMAN H., WAHLSTER W., Eds., *Spinning the Semantic Web*, The MIT Press, Cambridge, Massachusetts, 2003.

[GRU 93] GRUBER T. R., « *Formal Ontology in Conceptual Analysis and Knowledge Representation* », chapitre Toward Principles for the Design of Ontologies Used for Knowledge Sharing, Kluwer Academic Publishers, 1993.

[HAA 01] HAARSLEV V., MÖLLER R., « RACER System Description », *First International Joint Conference on Automated Reasoning, IJCAR'2001*, vol. 2083 de *Lecture Notes in Computer Science*, Springer-Verlag, 2001, p. 701–706, http ://www.racer-systems.com/.

[HOR 04] HORRIDGE M., KNUBLAUCH H., RECTOR A., STEVENS R., WROE C., « A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools. Edition 1.0 », University Of Manchester, 2004.

[HUL 04] AL HULOU R., NAPOLI A., NAUER E., « Une mesure de similarité sémantique pour la classification de documents par le contenu », EUZENAT J., CARRÉ B., Eds., *Langages et Modèles à Objets (LMO'04)*, vol. 10 de *RSTI - L'objet*, p. 217–230, Hermes, Paris, 2004.

[OCH 00] OCHSENBEIN F., « Astronomical Catalogues and Tables Adopted Standards version 2.0 », 2000, http ://vizier.u-strasbg.fr/doc/catstd.htx/.

[RIC 05] RICHARD A., « Construction d'une ontologie de descripteurs en astronomie à partir de tables de données », Mémoire de DEA, Juin 2005, LORIA.

[SAF 04] SAFAR B., KEFI H., REYNAUD C., « OntoRefiner, a user query refinement interface usable for Semantic Web Portals », *Application of Semantic Web Technologies to Web Communities Workshop, 16th European Conference on Artificial Intelligence*, 2004.

[STA 04] STAAB S., STUDER R., Eds., *Handbook on Ontologies*, International Handbooks on Information Systems, Springer, 2004.

[USC 95] USCHOLD M., KING M., « Towards a Methodology for Building Ontologies », *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conduction with IJCAI-95*, 1995.

[USC 96] USCHOLD M., GRUNINGER M., « Ontologies : Principles, Methods and Applications », *Knowledge Engineering Review*, vol. 11, n° 2, 1996, p. 93–155.