N° ordre /2006- M/IN

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie

Houari Boumediene

U.S.T.H.B

FACULTE D'ELECTRONIQUE & INFORMATIQUE

Mémoire présenté pour l'obtention du diplôme de :

MAGISTER EN INFORMATIQUE

Spécialité : Intelligence Artificielle et Base de Données Avancées

Par : GUEBAILI Ratiba

Thème

Réutilisation des graphes conceptuels dans le Web Sémantique avec introduction d'information floue

Soutenu : 08 Juillet 2006.

Devant le jury composé de:

Pr.	AHMED NACER Mohamed	Président
Pr.	MOKHTARI AÏCHA	Directrice de thèse
Dr	ALIMAZIGHI Zaia	Examinatrice
Dr	BELKHIR Abdelkader	Examinateur

PROMOTION : 2004/2005

Dédicaces

Dieu seul sait combien j'ai d'amour et de respect à la prunelle de mes yeux, ces deux êtres qui m'ont donné la vie et tout l'Amour et la tendresse qui existent sur terre : « PAPA » et « MAMAN » que Dieu me garde ma Maman et m'aide pour pouvoir rendre un peu de leur amour et accueil mon « PAPA » dans son Paradis.

Après eux à ma Seconde mère, et mon deuxième sourire dans la vie : ma sœur aînée « NANA » pour tout ce qu'elle m'a donné et me donne.

A mes frères : Djamel , Abdelhak et Réda.

A mes sœurs : Chahra Zed, Sabah, Zahra et Leila.

A mes beaux-frères et belles-sœurs.

A mes neveux Imad – Eddine, Hethem Tarek Aziz et Sid Ali.

A mes nièces Hiba, Sarah, Esma Khadidja, Chaîma et ma chère petite Khawla.

A mes amies sœurs Sadjia, Rachida et Nawel.

A mes amies Ilhem, Chahira, Lila, Sara, Ismahane, Mounia, Lila et Lynda.

A ma Directrice de thèse Mme Aissani – Mokhtari et ma Co- Directrice Mme Akli- Astouati

A ceux que je n'ai pas cité et qui ont un jour fait partie du train de ma vie, de près ou de loin.

Un grand Merci

Ratiba

Remerciements

Je tiens à remercier M^{me} AKLI Karima, chargée de cours à l'U.STHB, pour m'avoir consacré autant de temps et d'efforts pendant ces deux années. Je remercie aussi M^{me} Mokhtari Aîcha, professeur à l'USTHB pour sa confiance en acceptant d'être Encadreur de ce travail et pour leurs remarques qui ont permis d'améliorer la qualité de ce mémoire. Je tiens également à les remercier pour l'honneur qu'elles m'ont fait.

Je remercie Mr AHMED NACER Mohamed de m'avoir fait l'honneur de présider le jury

Je remercie Mme ALIMAZIGHI et M^{r} BELKHIR qui m'ont fait l'honneur de participer à mon jury, je leur en suis reconnaissante.

Je remercie également tous les enseignants de l'USTHB pour leur sacrifice, surtout les enseignantes : M^{me} Khellaf, M^{me} Ighilaza et M^{elle} Mamache pour leur soutient moral.

Merci et mille mercis pour ma famille, qui m'a toujours soutenu et qui m'a permis de mener à bien mes études.

Merci à tous ceux qui m'ont aidé de près ou de loin.

Résumé

Dans le cadre de la représentation de connaissances, on s'intéresse au support d'un graphe conceptuel. Ce dernier définit un vocabulaire de base avec lequel on représente sur un domaine des connaissances, dites « terminologiques », ce qui à un niveau plus philosophique on appellerait une ontologie du domaine. Ce support de graphe conceptuel a été étendu pour constituer les graphes conceptuels flous. Cela est fait par une redéfinition des types de concepts et de relations. Rapporter une telle représentation pour pouvoir l'utiliser au sein du web, tout en associant une sémantique aux ressources, peut être fait si un passage des graphes conceptuels flous vers un des langages du web sémantique RDF(S) (Resource Description Framework) est mis en œuvre en précisant les règles nécessaires. Le travail présenté aura comme objectif de proposer quelques solutions pour la représentation des différents types de concepts et de relations en RDF(S), en incluant la notion d'imprécision et de floue.

Mots clés : Web Sémantique, concepts, relation conceptuelle, RDF, RDF(S), Graphes conceptuels, imprécision, information floue.

Sommaire

Chapitre III : Les graphes conceptuels et les graphes Conceptuels flous

Chapitre IV

Passage Graphes conceptuels flous vers RDF(S)

et vice versa

Introduction Générale

Introduction générale

La description et la définition de concepts est un problème fondamental de l'Intelligence Artificielle, et plus spécialement de la représentation des connaissances. Un concept comme structure symbolique possède un sens appelé « intension du concept» et représente un ensemble d'objets appelé « extension du concept ».

L'un des objectifs principaux de la représentation de concepts est de pouvoir représenter des concepts en machine et de produire des raisonnements logiques sur ces concepts.

Dans un but de représentation et de raisonnement sur les intensions, des formalismes logiques sémantiquement bien définis sont utilisés. Leur caractéristique nécessaire se situe au niveau de la décidabilité, qui garantit que l'on dispose de procédures effectives de calcul.

On peut distinguer deux sortes de tâches de raisonnement sur les concepts, les tâches de raisonnement sur les concepts et les tâches de raisonnement assertionnelles.

Les tâches de raisonnement sur les concepts sont des tâches de raisonnement n'utilisant que les intensions, comme par exemple, déterminer si un objet satisfait une intension et appartient ainsi à l'extension d'un concept.

Par contre, les tâches de raisonnement assertionnelles sont des tâches de raisonnement utilisant à la fois des intensions et les extensions des concepts dans le cas où on dispose d'une base d'objets, reliés entre elles par des relations (appelées aussi rôles).

Nous appelons « ontologie » la partie terminologique, qui comprend les définitions de classes et de rôles, ainsi que des axiomes (des assertions entre les intensions). Une ontologie définit un vocabulaire symbolique conceptuel.

La description de concepts symboliques s'est aussi enrichie d'apports d'un formalisme général de représentation des connaissances, les Graphes Conceptuels.

Les Graphes Conceptuels sont un formalisme de notation graphique de la logique dans lequel le raisonnement peut se faire par projection de graphes. Sa forme la plus générale inclut la négation et a un pouvoir d'expression équivalent à la logique du premier ordre. Pour des raisons de décidabilité, le fragment le plus utilisé est celui des Graphes Conceptuels Simples, qui correspond au sous- fragment de la logique du premier ordre constitué des formules positives, existentielles et conjonctives. Ce sous- fragment a fournit un moyen de description de concepts simple et très lisible. Un concept est un graphe dont un noeud particulier représente l'objet dénoté. D'une façon plus générale, un concept n-aire peut être défini comme un graphe possédant « n » noeuds distingués.

Les graphes conceptuels sont un formalisme de représentation pivot entre la logique et le langage naturel. Or, ce dernier est connu par son caractère vague et imprécis du fait qu'il emploi des termes et des concepts qui ne peuvent être représentés en logique bivaluée. En d'autres termes, on ne peut associer seulement des valeurs de vérité « Vraie » ou « fausse » mais des valeurs intermédiaires comme « très vrai », « plus ou moins faux », etc.. peuvent exister. Pour le faire, une extension du modèle des graphes conceptuels a été proposée pour prendre en compte ce caractère flou, connu sous le nom de « Graphe Conceptuel Flou » [Cao01].

Ces résultats de la représentation de concepts en Intelligence Artificielle sont intéressants pour résoudre un problème actuel sur une application concrète de grande envergure : l'évolution d'un Web syntaxique vers un Web sémantique. Le défi de cette évolution est de transformer le World Wide Web actuel, entièrement tourné vers la présentation des documents à des utilisateurs humains, vers un Web dont le contenu serait compréhensible aux machines. La vision s'appuie sur l'utilisation d'un langage de description commune, permettant d'exprimer à la fois des ontologies, rassemblant un ensemble de définitions de concepts et de rôles, et d'exprimer des annotations et des métadonnées (par exemple sur les ressources du Web), utilisant le vocabulaire de ces ontologies.

Ce langage permettrait aux agents informatiques de comprendre les diverses métadonnées et annotations et de communiquer entre elles, en effectuant des raisonnements sur les concepts. Ce sont bien les problématiques fondamentales de la représentation de concept en Intelligence Artificielle.

Le langage qui semble aujourd'hui s'imposer est RDF (Resource Description Framework), proposé par le W3C (World Wide Web Consortium). Ce langage très simple possède à la fois une syntaxe graphique (ainsi qu'une syntaxe du type triplet, ou ressource-attribut-valeur) et correspond à peu près aux Graphes Conceptuels Simples.

Nous proposons dans ce travail, un moyen de passage qui permettra à travers un ensemble de règles de pouvoir utiliser les points forts des graphes conceptuels flous vers le web Sémantique. Les motivations derrière une telle proposition sont :

Réutilisation : Dans le monde de l'informatique, il n'existe pas de place pour le mot *abandonner* un programme ou un logiciel déjà créé. En effet, il serait intéressant d'exploiter des applications déjà développées à partir des graphes conceptuels vers RDF(S) afin de les réutiliser facilement dans le cadre du web sémantique. En d'autres termes, il existe des éditeurs de graphes conceptuels, sur lesquels on applique des opérations afin de produire de nouvelles connaissances. Il serait alors possible de réutiliser ces résultats dans le web sémantique grâce au passage graphe conceptuel (GC) vers RDF(S).

Méconnaissance de RDF : Lorsqu'on est habitué à écrire des prédicats ou à dessiner des graphes conceptuels représentant des expressions du langage naturel, on peut facilement s'intégrer dans la communauté du web sémantique sans avoir à apprendre le langage RDF(S) par le biais d'interface automatique réalisant ce passage.

Ouverture : Les graphes conceptuels sont munis d'un certain nombre d'opérations permettant la déduction de nouvelles connaissances. Réaliser le passage vers RDF(S) peut permettre de nouvelles recherches visant la réalisation d'un tel processus de déduction dans le web pour une plus grande ouverture.

Prise en compte de l'imprécision : Dans le cadre des recherches réalisées par l'équipe de l'INRIA Sophia Antipolis au sein de son projet ACACIA [Sav03], une extension de RDF(S) pour la représentation de connaissances contextuelles basées sur le modèle des graphes conceptuels [Del03] a été proposée. Les déclarations dans cette proposition sont toutes certaines, dans le sens où l'information est connue et précise. Il n'existe pas de moyen pour exprimer des déclarations comme « il est vrai que ….. », « Peut-être que … », ... ou encore le caractère vague ou imprécis des concepts et des relations. L'objectif de ce travail est d'ajouter ce caractère, i.e., se situer au niveau des concepts en ajoutant une valeur supplémentaire reflétant le caractère imprécis ou incertain de l'information.

Dans le présent travail, on commencera par donner un bref aperçu sur le web sémantique en montrant l'importance des ontologies et les métadonnées pour exprimer cette sémantique. Puis, une énumération des besoins et des objectifs d'un langage d'ontologie du web est présentée.

Le chapitre 2, va être consacré à une étude détaillée des deux langages du web sémantique pour la définition des métadonnées, à savoir RDF et RDF(S) après avoir montré les lacunes du langage XML.

L'étude des graphes conceptuels et leur extension floue feront l'objet du troisième chapitre, qui mettra le point sur la classification des concepts et des relations conceptuelles.

Le chapitre quatre sera consacré à notre proposition qui consiste à définir des règles de passage de RDF(S) vers les graphes conceptuels (ainsi que leur version floue) et vice versa. Un exemple d'utilisation de notre proposition dans un processus de négociation sera présenté.

On terminera par une conclusion générale et des perspectives à ce travail.

Chapitre 1

Chapitre I : Généralités sur le Web Sémantique

I. Introduction

L'expression Web sémantique, due à Tim Berners-Lee [Ber01] au sein du World Wide Web Consortium (W3C), fait d'abord référence à la vision du Web de demain comme un vaste espace d'échange de ressources entre être humains et machines permettant une exploitation, qualitativement supérieure, de grands volumes d'informations et de services variés. Espace virtuel, il devrait voir, à la différence du Web que nous connaissons aujourd'hui, les utilisateurs déchargés d'une bonne partie de leurs tâches de recherche, de construction et de combinaison des résultats, grâce aux capacités accrues des machines à accéder aux contenus des ressources et à effectuer des raisonnements sur ceux-ci.

Le Web actuel est essentiellement syntaxique, dans le sens où la structure des documents (ou ressources au sens large) est bien définie. Par contre, leur contenu, interprétable par l'humain, reste quasi inaccessible aux traitements machines. La nouvelle génération de Web – Le Web sémantique- a pour ambition de lever cette difficulté. Les ressources du Web seront plus aisément accessibles aussi bien par l'homme que par la machine, grâce à la représentation sémantique de leurs contenus.

Le Web sémantique, concrètement, est d'abord une *infrastructure* pour permettre l'utilisation de connaissances *formalisées* en plus du contenu informel actuel du Web, même si aucun consensus n'existe sur jusqu'où cette formalisation doit aller. Cette infrastructure doit :

- Permettre d'abord de localiser, d'identifier et de transformer des ressources de manière robuste et saine tout en renforçant l'esprit d'ouverture de web avec sa diversité d'utilisateurs.
- S'appuyer sur un certain niveau de consensus portant, par exemple, sur les langages de représentation ou sur les ontologies utilisées.
- Contribuer à assurer, le plus automatiquement possible, l'interopérabilité et les transformations entre les différents formalismes et les différentes ontologies.
- Faciliter la mise en œuvre de calculs et de raisonnements complexes tout en offrant des garanties supérieures sur la validité.

- Offrir des mécanismes de protection (droits d'accès, d'utilisation et de reproduction), ainsi que des mécanismes permettant de qualifier les connaissances afin d'augmenter le niveau de confiance des utilisateurs.

Mais restreindre le Web sémantique à cette infrastructure serait trop limitatif. Ce sont les applications développées sur cette infrastructure qui font et feront vivre cette vision et qui seront, d'une certaine manière, la preuve du concept. Bien sûr, de manière duale, le développement des outils, intégrant les standards du Web sémantique, doit permettre de réaliser plus facilement et à moindre coût des applications ou des services développés aujourd'hui de manière ad-hoc.

D'un point de vue architectural, le Web sémantique est une infrastructure s'articulant autours de trois composantes majeures :

- Les *métadonnées*, qui permettent de qualifier les flux informels du Web ;
- Les *ontologies*, destinées à représenter, dans un formalisme opératoire, les connaissances utiles à la méta-description ;
- Les *systèmes de raisonnement et de calcul*, qui combinent des données issues de sources hétérogènes pour produire de nouvelles informations, susceptibles d'être plus précises, plus adéquates et moins redondantes.

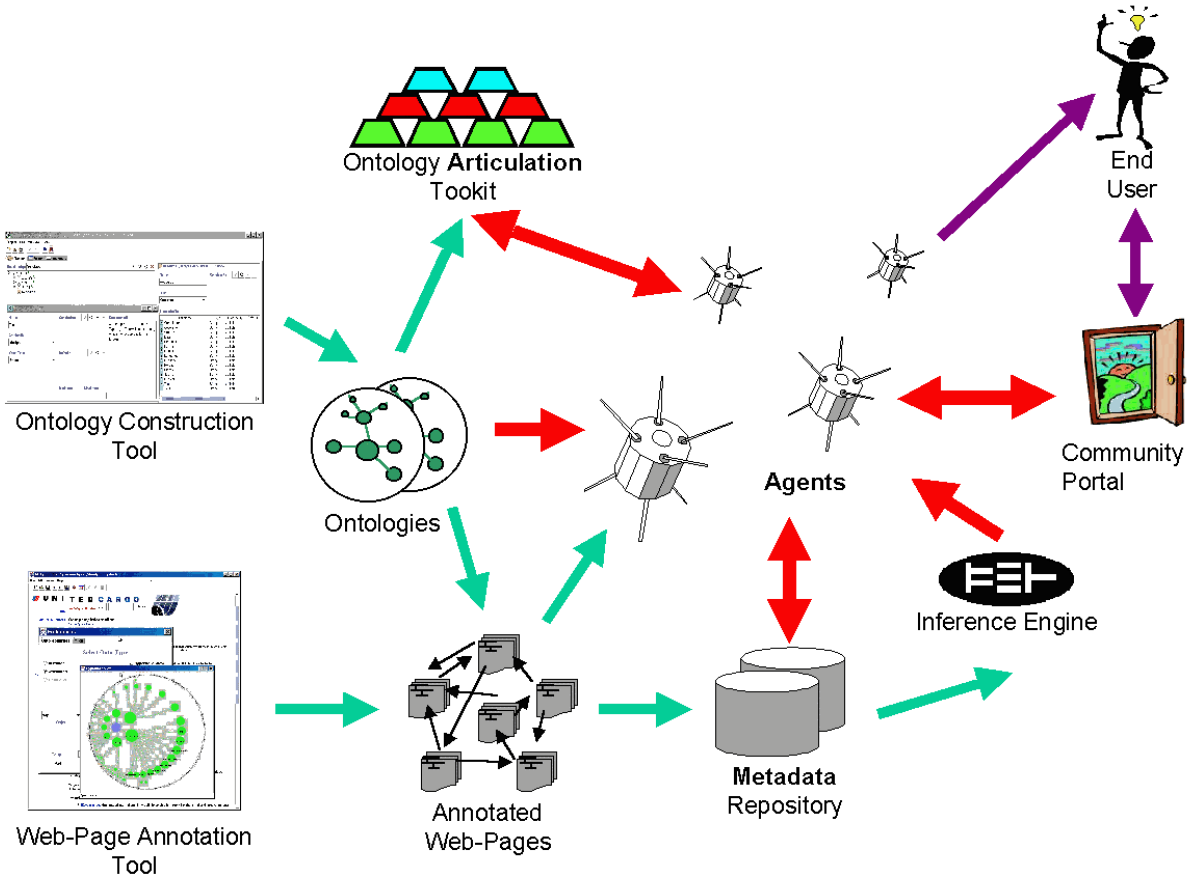

Figure 1. Utilisation des métadonnées dans le web Sémantique

Le schéma ci-dessus [Hac03] donne une vision de l'utilisation des méta-données sur le web sémantique. Des pages web sont annotées à partir de connaissances disponibles dans une ou plusieurs ontologies qui ont pour objectif de normaliser la sémantique des annotations, et ces annotations, regroupées en entrepôts de méta-données deviennent utiles pour des agents de recherche d'information, faisant ou non appel à des moteurs d'inférence permettant de déduire de nouvelles connaissances formelle des annotations.

II. Ingénierie ontologique

L'ingénierie ontologique a émergé de l'ingénierie des connaissances. Cette dernière est considérée comme le domaine de prédilection du développement d'expertise en conception de systèmes à base de connaissances [Miz04]. Malgré le fait que l'ingénierie des connaissances ait contribué à accroître cette expertise en l'organisant dans une perspective computationnelle, certains membres de la communauté de l'IA ont éprouvé le besoin de passer à une ingénierie s'appuyant plus solidement sur les fondement théoriques et méthodologiques améliorant ainsi la conception de systèmes intelligents.

Une ontologie fournit un système de concepts fondamentaux c'est à dire un système de connaissances qui constitue les connaissances d'arrière plan d'une base de connaissances. Une ontologie offre une conceptualisation du monde cible ainsi qu'une base solide sur laquelle des bases de connaissances partageables plus largement utilisable qu'une base de connaissances conventionnelles.

Le rôle d'une ontologie envers une base de connaissance est de donner des définitions de concepts utilisés dans la représentation de connaissances ; C'est aussi de spécifier les contraintes entre les concepts, afin de rendre la base consistante et transparente, deux propriétés nécessaires pour pouvoir partager et réutiliser la connaissance.

L'ingénieur ontologique considère les différentes sources de connaissances qui ont été classées de 3 façons [Miz04] :

- Les connaissances de sens commun ;
- L'expertise acquise par des expériences spécialisées ;
- Les connaissances théoriques.

Une ontologie devrait être développée par une communauté dont les membres partage la nécessité de posséder une ontologie commune. Les développeurs d'ontologie doivent se rattacher à une ontologie de haut niveau capable de les guider pour construire des ontologies « raisonnables ».

Deux types d'ontologie sont distingués [Miz04] :

- Une ontologie orientée web sémantique qui est un vocabulaire compréhensible par un ordinateur et qui définit la signification des métadonnées. Ce type d'ontologie peut-être qualifié d'ontologie de surface puisqu'elle ne traite pas nécessairement de la structure conceptuelle profonde du monde cible.

- Une ontologie orientée concept qui traite des concepts fondamentaux du monde cible et qui demandent à être examinés en profondeur.

Dans la suite de notre travail, nous nous sommes intéressé au premier type d'ontologies.

III. Ontologie orientée concept

Tout programme informatique manipule, à travers des symboles, les objets du domaine modélisé. L'ensemble de ces objets correspond à ce qui est appelé *référentiel* dans le domaine des systèmes d'information. Pour un domaine mettant en œuvre des connaissances complexes sur lesquelles on veut effectuer des traitements intelligents, le programme élaboré est un système à base de connaissances. Cette base répertorie, entre autres, et de la même façon que dans les systèmes d'information, les concepts du domaine hiérarchiquement organisés dans une « ontologie ».

Les ontologies sont apparues au début des années 90 dans la communauté ingénierie des connaissances, dans le cadre des démarches d'acquisition pour les systèmes à base de connaissances.

Elles cherchent à s'appuyer sur des modélisations de ressources du web à partir de représentations conceptuelles des domaines concernés et, ceci dans le but de permettre à des programmes de faire des inférences dessus. Les recherches à leur sujet sont donc indispensables.

Une fois construite et acceptée par une communauté particulière, une ontologie doit en effet traduire un ***consensus explicite*** et un certain ***niveau de partage***, deux aspects essentiels pour permettre l'exploitation des ressources du Web par différentes applications ou agents logiciels. D'autres part, la formalisation, autre facette des ontologies, est nécessaire pour qu'il soit possible de raisonner automatiquement dessus afin de décharger les utilisateurs d'une partie de leur tâche d'exploitation des ressources du Web. Les ontologies servent alors [Cha03] :

(1) Comme représentation pivot pour l'intégration de ressources de données hétérogènes.

(2) Pour décrire les services Web et, en général, partout où il va être nécessaire d'appuyer des modules logiciels sur des représentations sémantiques nécessitant un certain consensus.

III.1. Définition

Une définition d'une ontologie donnée dans [Cha03] est la suivante:

« *Ensemble des objets reconnus comme existant dans le domaine. Construire une ontologie c'est aussi décider de la manière d'être et d'exister des objets ».*

Dans cette définition, les objets ne sont pas pris dans un sens informatique mais comme objets du monde réel que le système modélisé.

Dans [Fur02], on retrouve un ensemble de définitions que l'on énumère comme suit :

- *« Une ontologie implique ou comprend une certaine vue du monde par rapport à un domaine donné. Cette vue est souvent conçue comme un ensemble de concepts – e.g. entités, attributs, processus-, leurs définitions et leurs inter-relations. On appelle cela une conceptualisation ».*

- *« Une ontologie peut prendre différentes formes mais elle inclura nécessairement un vocabulaire de termes et une spécification de leur signification ».*

- *« Une ontologie est une spécification rendant partiellement compte d'une conceptualisation »*

Ou encore, pour [Grû95], « *une ontologie est une description d'entités et leurs propriétés, relations, contraintes, comportement ».*

Un complément de cette définition est donné dans [Sch97] « *une ontologie est un ensemble de spécifications de concepts compréhensibles par une machine ».*

Mais, la définition la plus utilisée est donc la plus consensuelle revient à Gruber [Gru93] *« Une ontologie est une spécification explicite d'une conceptualisation ».*

Grâce à ces définitions on peut conclure que :

- Une ontologie est bien une conceptualisation, entendons par là que l'on y définit des concepts ;

- Devant être par la suite utilisée dans un artefact informatique dont on veut spécifier le comportement, l'ontologie devra également être une théorie logique pour laquelle on précisera le vocabulaire manipulé ;

- Enfin, la conceptualisation étant spécifiée parfois de manière très précise, une théorie logique ne peut pas toujours en rendre compte de façon exacte. Elle ne peut assumer la richesse interprétative du domaine conceptualisé dans une ontologie et ne le fait donc que partiellement.

III.2. Caractéristiques

A partir des définitions proposées jusque-là pour les ontologies, quatre grands types de caractéristiques nous permettent de préciser ce qui peut être représenté dans une ontologie ainsi que dans le processus de modélisation.

- ***Le type d'ontologie.*** Les ontologies peuvent être classifiées en fonction de deux dimensions [Fur02]: leur niveau de détail et leur niveau de dépendance par rapport à une tâche particulière. Par contre dans [Gua97], on distingue plusieurs niveaux dans les ontologies :

- (1) *L'ontologie du domaine*. Les ontologies liées à un domaine particulier sont de deux types :
    - Elles contiennent le vocabulaire spécifique à un domaine bien défini et sont des spécialisations d'une ontologie de haut niveau ;
    - Il s'agit d'ontologies de tâche qui contiennent l'ensemble des tâches réalisées dans un domaine donné.

(2) *L'ontologie générique* ou qui se veut comme telle, qui repère et organise les concepts les plus abstraits du domaine. Ces ontologies, dites de haut niveau, contiennent les concepts généraux, commun à tous les domaines (temps, espace, objet, évènement). Les travaux de John Sowa et de Nicola Guarino tentent de formaliser une telle ontologie « universelle ».

(3) *L'ontologie d'une méthode de résolution* de problème où le rôle joué par chaque concept dans le raisonnement est rendu explicite ;

(4) *L'ontologie d'application* qui se veut une double spécialisation d'une ontologie du domaine et d'une ontologie de méthode ;

(5) *L'ontologie de représentation* qui repère et organise les primitives de la théorie logique permettant de représenter l'ontologie.

Ces différents niveaux sont récapitulés dans la figure 2. Par conséquent, une ontologie peut être vue comme une théorie qui distingue les concepts particuliers, c'est-à-dire les objets concrets, physiques, les évènements, les régions, …etc., et les concepts universels c'est-à-dire les propriétés, rôles, états, etc.

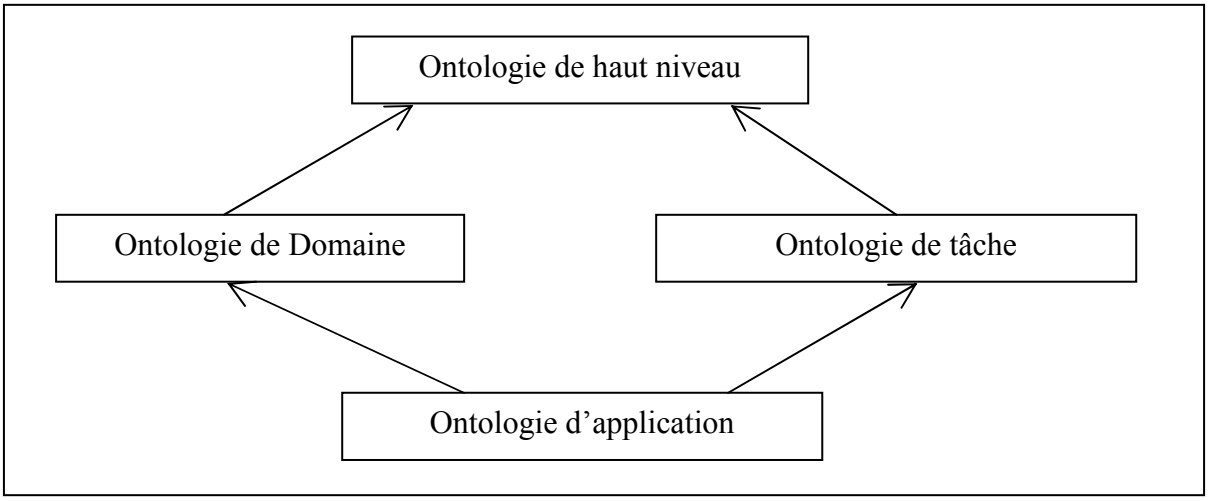

Figure 2. Différents types d'ontologie selon leur degré de dépendance vis-à-vis d'une tâche particulière ou d'un point de vue.

- Les propriétés. Une ontologie est non seulement le repérage et la classification des concepts mais c'est aussi des caractéristiques qui leur sont attachées et qu'on appelle ici des propriétés. Ces dernières peuvent être valuées.

- La relation « is-a ». La relation de subsomption is-a qui définit un lien de généralisation est utilisée pour structurer les ontologies. Cette relation qui permet formellement l'héritage de propriétés est un choix qui s'impose depuis ARISTOTE. Elle doit être complétée par d'autres relations pour exprimer la sémantique du domaine.

- Les autres relations. Les relations unissent les concepts ensemble pour construire des représentations conceptuelles complexes qui vont être autant de connaissances nécessaires à l'ontologie que l'on construit. Si la connaissance construite correspond à un concept dans le monde modélisé, celui-ci est dit ***défini***. A l'opposé, des concepts insérés dans l'arborescence de l'ontologie sont dits ***primitifs***. Par exemple, si l'on définit l'appendicite comme une ***inflammation localisée -sur l'appendice***, c'est un concept dit défini. Dans l'exemple précédent, localisée-sur est une relation binaire qui se définit par les concepts qu'elle relie et par le fait qu'elle est, comme les concepts, insérée dans une hiérarchie, ici de relations : c'est un concept primitif.

III.3. Le rôle des ontologies

La communauté d'intelligence artificielle utilise les ontologies pour d'autres raisons dont deux principales :

III.3.1. Modularité et réutilisabilité des connaissances

Les ontologies sont surtout utilisées pour la représentation de connaissances et l'application de raisonnements sur ces dernières. Cependant, une ontologie possède des caractéristiques qui, au-delà de cette représentation, favorise la réutilisation et le partage des données. Déjà en 91, Thomas Gruber [Gru91] insistait sur le rôle que pouvaient tenir les ontologies pour favoriser la modularité et la réutilisabilité dans les systèmes informatiques.

En effet, ces ontologies permettent l'étude de conceptualisations, indépendamment du formalisme choisi pour les représenter [Val96] et doivent être définies indépendamment du langage utilisé pour la programmation des applications, de la plate-forme utilisée et des protocoles de communication (protocoles réseaux). Cependant, Des difficultés techniques apparaissent. Ils sont soulignés [Gru95]. Pour l'auteur les systèmes à base de connaissances mettent en place des techniques d'interopérabilité basées sur la communication et les opérations à partir de représentations formelles de connaissance. Souvent ils peuvent être comparés à des agents qui négocient et échangent des connaissances. Trois niveaux de convention sont alors nécessaires :

- Le format de représentation du langage ;
- Le protocole de communication des agents;
- La spécification du contenu du vocabulaire partagé.

Et c'est surtout sur ce dernier point que les ontologies peuvent jouer un rôle intéressant.

III.3.2. Communication

Il existe trois types de communication dans un projet : communication homme- homme, homme- système ou entre différents modules du système. Ces trois types possèdent tous des caractéristiques particulières qui engendrent certains problèmes auxquels les ontologies peuvent apporter des solutions.

La communication entre humain pose surtout des problèmes quand les acteurs de cette communication ne sont pas du même domaine et ne parlent donc pas forcément le même langage. La réutilisation, le partage de connaissance et d'ontologies, suppose que plusieurs utilisateurs soient d'accord sur les ontologies partagées. C'est ce que Fensel et al [Fen98]. Nomment un « ontogroup ». Philippe Martin [Mar95] propose d'aider les spécialistes de l'ingénierie de connaissance en utilisant la terminologie définie dans WordNet comme base de la communication, car c'est un standard. Dans le domaine pédagogique c'est la communication entre auteurs et informaticiens qui est parfois difficile, d'où l'intérêt d'utiliser

des ontologies dans les environnements auteur pour la définition d'un vocabulaire convivial et précis dans la définition des tâches pédagogiques [Ike99]. L'ontologie jour alors le rôle d'un méta- modèle.

Une fois que les acteurs humains d'un projet sont d'accord sur une ontologie, la communication avec le système se fait naturellement, en utilisant cette ontologie. De plus, l'adaptation des ontologies à la description de textes en langage naturel, semi- structurés [Kle00] améliore la communication dans le sens machine – homme.

Les ontologies peuvent également être utilisées pour harmoniser la communication entre différentes applications ou entre différents agents. Cette idée, également sous-jacente dans les publications de Gruber [Gru95], repose souvent sur une ontologie du domaine. Pourtant Chen et Mizoguchi [Che99] veulent aller loin en dotant les agents d'une connaissance sur une ontologie de tâche indépendante du domaine.

Pour synthétiser, on peut dire que si le rôle principal d'une ontologie est de favoriser le partage et la réutilisation de la connaissance, il faut cependant distinguer plusieurs types d'utilisation qui entraînent des besoins différents [Val96] :

- Une ontologie peut être utilisée comme un répertoire dans lequel on stocke et organise des connaissances et des informations. Elle peut concerner des données simples, standardisées dans un domaine particulier ou bien des données distribuées ;
- En acquisition de connaissance, les ontologies rassemblent les définitions des termes d'un domaine ce qui permet à plusieurs acteurs de communiquer sans ambiguïté ;
- L'ontologie doit également contenir certaines définitions qui permettent d'assurer la consistance de la base de connaissance et son utilisation correcte ;
- Les ontologies se justifient souvent par la volonté de réutiliser la connaissance pour la construction de nouvelles applications ;
- Enfin, une ontologie peut être utilise comme la base d'un langage de représentation des connaissances.

IV. Méta-données et annotations

Un des grands principes du Web sémantique est qu'il est nécessaire d'associer aux ressources du Web des informations exploitables par des agents logiciels afin de favoriser l'exploitation de ces ressources [Pri03].

Associer par exemple une notice comprenant des champs : Auteur, Date de création, Date de modification, Mots-clés, à une page Web permettrait à un agent logiciel de les utiliser avec leur sens au lieu de considérer cette page comme comprenant un texte seulement. Ces informations sont les métadonnées.

Une méta-donnée est « une données sur une donnée ». Cette définition est un peu vague voire ambiguë. Elle est comprise de manière différente par différentes communautés qui conçoivent, créent, décrivent, préservent et utilisent des systèmes d'information et des ressources. Par exemples, dans certains cas la donnée sur laquelle la métadonnée porte est considérée comme ayant le même statut de donnée formalisée, traitable par un système informatique. Dans d'autre, la donnée n'est qu'interprétable par un être humain, et seule la méta-donnée en permet le traitement automatique.

Une annotation es à la base une note critique ou explicative accompagnant un texte, et par extension, une quelconque marque de lecture portée sur un document, que celui-ci soit textuel ou de type image. Elle permet d'associer des notes de lecture aux documents, de partager l'information, d'effectuer des tâches rédactionnelles en groupe, etc. L'annotation sémantique à partir d'ontologies semble être la meilleure approche pour partager et exploiter l'information sur le web. Les outils d'annotations visent à améliorer la communication et l'interopérabilité sur le web.

V. Objectifs opérationnels d'un langage d'ontologie du web

On cite dans ce qui suit huit (08) objectifs opérationnels, tirés de [Hel02] :

1. Ontologies partagées. Plusieurs sources de données peuvent accéder à une même ontologie dans un but de partage de connaissances. Une ontologie offre des termes ainsi que des descriptions formelles de ces termes. Les sources de données qui convergent vers la même ontologie, acceptent d'utiliser les mêmes termes avec bien sur leurs sens déjà définis.

2. Evolutivité des ontologies. Les ontologies sont sujettes à de fréquentes modifications de tous types (corrections d'erreurs, changement d'existant, nouvelle modélisation des besoins), ce qui nécessite que chaque source doit préciser quelle version utiliser.

3. Interopérabilité entre ontologies. Le même concept peut être modélisé différemment par diverses ontologies. Il est donc important que le langage ontologique offre des primitives pour relier ces différentes représentations, permettant ainsi de convertir les données sous différentes ontologies à « un réseau d'ontologie ». Bien que le partage et l'extension d'ontologies permettent un certain degré d'interopérabilité entre les différents domaines, il

existe de nombreux cas où une information peut être modélisée de plusieurs façons, reflétant la différence dans les points de vues.

4. Détection des aberrations. Des incohérences et des contradictions peuvent évidemment exister. Ceci est dû au nombre de données que le Net peut supporter. Pour éviter que des agents ne combinent des informations contradictoires ou fausses, il est important que les aberrations puissent être détectées automatiquement. Exemple, dans le cas où les données sont décentralisées avec absence de l'autorité de contrôle.

5. Equilibre entre expressivité et capacité opératoire. Le langage doit être en mesure d'exprimer une large variété de connaissances mais doit aussi offrir, pour des raisons d'efficacité, des sens pour raisonner.

6. Facilité d'emploi. Le langage doit être facile à comprendre avec une syntaxe et une sémantique claire.

7. Syntaxe XML. Le langage devrait pouvoir être sérialisés sous format XML afin de bénéficier de tout ce qui a été acquis jusqu'à présent avec l'utilisation d'XML.

8. Internationalisation. Le langage doit permettre le développement d'ontologies multilingues et offrir potentiellement différentes vues de l'ontologie qui sont appropriées pour les différentes cultures.

VI. Conclusion

Le web apparaît aujourd'hui partout dans le monde comme une plus large base de ressources à accès distribué. Elle n'est pas moins volatile, incomplète, hétérogène et même pleine de contradiction. La recherche d'information sur ce web, par des moteurs de recherche, a vite montré ses limites et a encouragé l'utilisation des métadonnées dont les termes employés appartiennent à une ontologie. C'est ainsi que c'est succédé de nombreuses propositions pour l'ajout des métadonnées. Ce qui permettra d'ajouter au web la sémantique qui lui manquait.

Modéliser des métadonnées n'implique pas nécessairement un modèle de données complexes. A titre d'exemple, le Dublin Core [Dub00] se limite à un ensemble très restreint de quinze (15) champs alphanumérique (Title, Author, …etc.). Néanmoins, malgré son intérêt, le Dublin Core reste extrêmement limité pour décrire le contenu d'un document.

Pour obtenir une description manipulable du contenu sémantique d'un texte, un tel modèle est évidemment insuffisant. En réponse à un tel besoin, le W3C (Word Wide Web Consortium) a stabilisé dès 1999, la spécification RDF(S). Cette approche fait actuellement l'objet de beaucoup d'intérêt par la communauté du Web. L'étude de ce langage (RDF) et son extension RDF(S) se fera dans le chapitre suivant.

Chapitre 2

Chapitre II : Les Langages pour le Web Sémantique

I. Introduction

Un jeu de méta-données, plus ou moins réduit suffit pour une recherche d'information en associant un simple langage de programmation. Par contre, pour des travaux plus complexes mettant en œuvre des architectures sophistiquées comme l'exploitation de ressources hétérogènes ou la prise en compte de la sémantique, un langage adéquat s'impose. Il doit être plus expressif et plus formel.

II. Les besoins d'un langage opérationnel du web

Avant de citer les différents besoins opérationnels, on va présenter un ensemble de cas d'utilisation des ontologies, justifiant quelques besoins qui ne sont pas issus des objectifs opérationnels d'un langage dédié au web sémantique.

II.1. Cas d'utilisation

a. Les portails du Web

Le portail du web est un site Internet qui sert à collecter des informations sur un sujet commun (former des forums, recevoir des nouvelles, …etc.). Pour cela, il doit être le point d'entrée pour accéder à un contenu pertinent indexé par les membres de la communauté sous des thématiques. Mais, un simple index n'apporte pas de capacités de recherche de contenu comme voulu et attendu. Pour le faire, les portails Internet peuvent définir une ontologie pour la communauté. Cette dernière peut fournir une terminologie explicite pour décrire le contenu. Les inférences suggérées par l'ontologie peuvent être employées pour améliorer la qualité des recherches effectuées sur le portail.

Exemple : Des portails d'entreprise Sémantiques : Onto Broker® et Comma®

OntoBroker [Ont98] est la première tentative dans la mise en œuvre des technologies du web sémantique au Knowledge Management (KM). Son architecture se compose d'une interface d'interrogation, d'un moteur d'inférence et d'un collecteur de données sur le Web. Les formalismes d'interrogation définissent la notion d'instances, de classes, d'attributs et de valeurs.

Ce système a été mis en œuvre avec succès sur les scénarios d'usage suivants :

- Portails communautaires : Acquisition et partage de connaissances entre les communautés d'employés ;
- Annotations de documents.
- Gestion des ressources humaines.

Comma ® [Com00] est un projet subventionné par la communauté Européenne visant à développer et tester un environnement de gestion de mémoire d'entreprise.

Ce projet s'intéresse à deux scénarios :

- Aide à l'insertion d'un nouvel employé : Auto insertion de l'employé, en lui offrant une rapide compréhension de la politique et l'organisation de l'entreprise afin d'être opérationnel le plus rapidement possible et lui permettant ainsi de trouver, en lui suggérant, l'information dont il a besoin.

- Support de la veille technologique : Utiliser la mémoire d'entreprise pour assister l'identification et l'évaluation de technologies concernant l'activité de l'entreprise, et diffuser l'information pertinente aux personnes concernées et compétentes.

COMMA se distingue par son intégration de plusieurs technologies émergentes (XML, RDFS, Systèmes Multi-Agent, …etc.) apportant chacune des éléments de solution pour la réalisation, la gestion et l'exploitation des mémoires hétérogènes et distribuées.

b. Ressources Multimédias

Il est possible grâce aux ontologies d'écrire des descriptions sémantiques à des éléments de bibliothèques d'images, de son ou d'autres objets non textuels, permettant ainsi de servir à des procédés d'indexation et de recherche.

Les fonctionnalités de recherches à ce niveau doivent dépasser l'emploi de simples mots clés comme termes de recherches, en captant des connaissances additionnelles concernant un domaine qui peuvent être utilisées pour améliorer la récupération des images.

Cette technologie possède les fonctionnalités suivantes :

- Une taxonomie pour restreindre ou étendre des termes de recherches.
- Une structure tout/ partie utilisée par les recherches afin d'atteindre de meilleurs résultats.
- Une méthode d'expression de connaissances définitionnelles.
- Une représentation de connaissances par défaut.

Exemple. Soit la connaissance : 'une commode de la période fin d'époque géorgienne est traditionnellement fabriquée en acajou'. Cette information est cruciale pour une réelle exploitation de la sémantique d'une requête. Ainsi, une requête d'un utilisateur pourrait cibler des images de commodes à tiroir de la période fin d'époque géorgienne même si les annotations de ces images ne fournissent aucune indication sur le type de bois.

c. La calcul Omniscient

Le calcul omniscient est un nouveau paradigme de traitement informatique, permettant d'utiliser des ordinateurs dans notre environnement quotidien grâce aux caractéristiques des équipements non filaires,

qui sont portables et de petites tailles ce qui nécessite des architectures réseau supportant des configurations automatiques (sans intervention des utilisateurs du fait qu'ils sont des usagés ordinaires) et ad hoc.

La technologie clé des réseaux Ad Hoc est comment découvrir, décrire et annoncer des fonctionnalités offertes par les fournisseurs de services (fonctions offertes par une variété d'équipements comme le téléphone cellulaire, imprimante, équipement sonsor, etc.) qui dépendent largement de la standardisation.

Le but principal de ce type de traitement d'ubiquité est d'atteindre un grand niveau d'interopérabilité quelque soit la localisation géographique et les contraintes liées à cette dernière. En d'autres termes, le but est de faire collaborer des équipements distincts qui ne sont pas dédiés à travailler ensemble (buts de construction distincts, différents fabricants, etc.) malgré leur caractère mobile pour fournir des Web services.

L'utilisation des services nécessite : la découverte, la décomposition (réduction) et la composition de réseaux sans oublier la nécessité de représentation des informations relatives à la sécurité, le secret et la confiance.

Exemple

Supposant l'existence de deux personnes dans un super marché. Tous les deux possèdent par exemple un PDA (Personal Digital Assistant) sauvegardant chacun des pages Web. Ces pages sont présentées à l'utilisateur via une liste de sujets (topics). Dès que l'un des utilisateurs est à la porté de communication de l'autre, alors un *système d'informations spontané SIS* a lieu, et un espace virtuel constituant l'ensemble des pages web disponibles sur les deux PDAs est crée. L'utilisateur peut alors sauvegarder automatiquement les pages retrouvées après une recherche dans cet espace suivant un centre d'intérêt qu'il à déjà défini. Mais, une simple différence de définition d'un terme peut rendre cette recherche infructueuse surtout que le temps lié à la mobilité est restreint.

d. Documentation de développement

Les documents volumineux comme la documentation de développement, d'industrialisation et celle des tests, ont une structure hiérarchique différente entre elle, ainsi qu'un ensemble d'axes supposés chaînés l'ensemble de cette documentation.

Exemple. En aéronautique, on peut décrire les objets suivant leurs apparences.

Grâce aux ontologies, la construction d'un modèle de connaissances offrant la possibilité d'explorer le système d'information à travers ces objets consiste à représenter les associations entre ceux-ci, leurs propriétés, ainsi que les liens vers la documentation qui les décrit et les définit. Cela fait que, les ontologies et les taxonomies doivent être développées au même rythme que les objets physiques qu'elles décrivent.

Dans le domaine aéronautique, les utilisateurs typiques sont :

- Les ingénieurs de maintenance recherchant toute information relative à un sujet particulier (un type d'avion, une aile, les fournisseurs habituels, etc..).

- Les ingénieurs de développement recherchant les contraintes d'utilisation d'un sous-ensemble particulier.

Un cas d'utilisation courant de ce type d'ontologie, est la visualisation et l'édition de carte instantanée du système d'information en s'intéressant à un objet donné (qu'il soit lui-même une classe ou une instance, i.e, un cas général ou un cas particulier). Il résulte, alors des diagrammes d'activité.

Ce cas d'utilisation rencontre les besoins suivants :

- Expression des contraintes comme le tests de cohérence.

- Représentation en langage naturel.

- Des instances distinctes de classes (par exemple différents fournisseurs d'une même pièce).

- Des relations n-aires.

- Interface transparente utilisant standards XML.

e. Gestion des sites Internet institutionnels

Beaucoup d'organismes possèdent de nombreuses pages web destinées à proposer des offres de produits et des études de cas, des procédures coopératives, papiers blancs, et des descriptions de processus.

Les ontologies peuvent être exploitées pour indexer ces documents et apporter de meilleures solutions de recherche. Bien que les grandes organisations disposent en général de leur propre taxonomie pour organiser leurs informations, elle est souvent insuffisante. Une simple taxonomie se révèle limitée pour organiser des objets qui relèvent de plusieurs catégories. De plus, une recherche paramétrée est souvent plus efficace qu'une recherche par mots clés issus de ces taxonomies.

Les capacités de traitement ontologique d'un site Internet pourraient être exploitées ainsi :

- Application e – Commerce : Ces recherches peuvent être initialisées par l'expression des intérêts des clients dans une ontologie avec des personnes techniques qui peuvent décrire de tels articles. Cette recherche par mot clé est aussi inadéquate. Similairement, une taxonomie de domaine technique doit être d'utilisation limitée si le vendeur n'est pas familier avec le domaine. Comme exemple d'application e-commerce, l'application *OntoSeek* [Gar99]

- Pour les services de pages jaunes ou catalogues de produits, une représentation structurée des contenus couplés à des ontologies linguistiques améliore de manière notable le rappel et la précision des outils de recherche marchands. Le système *OntoSeek* a couplé une représentation des connaissances du

domaine en moyen d'un langage très limité, à une large ontologie linguistique multilingue pour une recherche de produits en langage naturelle multilingue. L'utilisateur, grâce à ce système, n'est pas supposé connaître le vocabulaire de codage des produits. L'ontologie linguistique peut s'exprimer avec les termes de son vocabulaire.

Les principaux choix d'architecture fonctionnelle d'OntoSeek :

- Usage d'une ontologie linguistique générique (représentant les produits).
- Flexibilité terminologique dans l'expression des requêtes.
- Assistance interactive pour la formulation de la requête.
- Application d'expertise : Dans ce champ, l'organisation de taxonomie de documents par contenu peut être utilisable. Mais, l'utilisation d'une taxonomie multiple doit être mieux adaptée depuis que des problèmes plus complexes tendent à mettre en cause des disciplines multiples.

Les ontologies institutionnelles pourraient nécessiter :

- Héritages multiples : Les classes sont souvent de multiples super classes ;
- Des relations tout/ partie. Par exemple : retrouver les étapes d'un projet par exemple.
- Une capacité d'enregistrement des relations entre les objets ou des présupposés ;
- Une interface transparente entre les ontologies du net : Utilisation du standard XML ;
- Une représentation indépendante de la langue afin d'assurer un plus grand partage d'informations

F. Agents intelligents

Le Web Sémantique peut doter les agents de la possibilité de comprendre et d'intégrer diverses sources d'information. Un exemple spécifique est la planification d'activités, qui peut prendre les préférences de l'utilisateur et les utiliser afin de planifier les activités de ce dernier. La tâche de planification des activités dépend de la richesse de l'environnement de service offert et des désirs de l'utilisateur durant la détermination de service. Le processus de planification doit être évalué et revu. Il peut être aussi consulté par le client afin de mieux approcher les préférences de ce dernier.

Ce type d'agent nécessite d'une part, des ontologies de domaine qui représentent les termes employés dans chacun des domaines (comme les termes employés dans une réservation d'une chambre d'hôtel ou pour passer une commande dans un restaurant) et d'autre part, des ontologies de service pour représenter les termes utilisés dans les services (termes de l'application).

Quand on construit les services, les informations peuvent venir de nombreuses sources, comme les portails (yahoo.com, citysearche.com, etc..), des sites à service spécifique (marriott.com, hyatt.com, etc.), des sites de réservation (reservation.com, etc.) ou bien le web en général.

Cela nécessite un nombre de domaines différents et ontologies de service :

- Utilisation et intégration de multiples ontologies séparées à travers différents domaines et services.
- Localisation distribuée d'ontologie à travers l'Internet.
- Différentes ontologies potentielles pour chaque domaine ou service (translation d'ontologie à travers le référencement).
- Représentation simple d'ontologie pour réaliser la tâche de définition et utilisation d'ontologies faciles.

II.2. Besoins opérationnels

Les cas d'utilisation et les objectifs opérationnels de l'ontologie imposent un certain nombre de besoins qu'un langage ontologique du Net doit remplir. Chaque besoin comprend une courte description et est justifié par un des cas d'utilisation ou un objectif opérationnel déjà cité :

1. ***Les ontologies sont des objets distincts :*** Les ontologies doivent être des objets qui ont des identifiants uniques, comme une référence URI. Il doit être possible d'identifier sans ambiguïté un terme d'une ontologie par une référence URI.

Motivation : C'est un besoin partagé par tous les objectifs opérationnels.

2. ***Extension d'ontologie explicite :*** Des ontologies peuvent définir d'autres ontologies explicitement, dans le but de réutiliser les termes tout en ajoutant de nouvelles classes et propriétés. Les extensions d'ontologies doivent être des relations transitives.

Motivation : C'est un besoin partagé par tous les objectifs opérationnels.

3. ***L'engagement dans une ontologie :*** les ressources doivent être capables de s'engager explicitement dans des ontologies spécifiques, en indiquant, précisément quels ensembles de définitions et hypothèses sont faites.

Motivation : Le partage d'une ontologie.

4. ***Métadonnée d'ontologie :*** Il doit être possible de fournir des méta- descriptions de chaque ontologie incluant : Son auteur, sa date de publication, …etc. Le langage doit fournir un jeu standard de métadonnées. Elles peuvent être ou non tirées du jeu d'éléments Dublin Core[1].

Motivation : C'est un besoin partagé par tous les objectifs opérationnels

[1] Dublain Core est un vocabulaire (ou ontologie) servant à décrire des œuvres littéraires ou artistiques. Elle permet de décrire toutes formes d'œuvres sur toutes les formes de support (articles de presse, photographies, sculptures,etc. Ce vocabulaire prévoit des propriétés basiques comme titre et l'auteur d'une œuvre…, jusqu'à des propriétés plus complexes gérant les notions de doits, de supports, etc.

4. ***Documenter les versions :*** Le langage doit pouvoir distinguer entre les versions et effectuer des comparaisons afin de préciser les compatibilités descendantes, ainsi que la possibilité de déclasser des termes.

Motivations : but partagé par tous les cas d'utilisation d'ontologies

5. ***Définitions de classes primitives :*** Le langage doit être capable d'exprimer des définitions complexes de classes. Cela comprend, sans s'y limiter, l'expression de sous classes et des combinaisons booléennes de classes.

Motivations : But partagé par tous les cas d'utilisation d'ontologie.

6. ***Primitives de définitions de propriétés :*** Le langage doit offrir les moyens d'exprimer les définitions de propriétés, de sous-propriétés, de domaine, la transitivité, les propriétés inverses et d'autres.

Motivations : But partagé par tous les cas d'utilisation d'ontologie.

7. ***Types de données :*** Des types de données basées sur les types de XML doivent être fournies par le langage.

Motivations : But partagé par tous les cas d'utilisation d'ontologie.

8. ***Equivalence des classes et des propriétés :*** Le langage doit offrir les moyens de déclarer que deux (02) classes ou deux propriétés sont équivalentes.

Motivations : But partagé par tous les cas d'utilisation d'ontologie.

9. ***Equivalence d'individu :*** Le langage soit fournir les moyens de déclaration qu'une paire d'identifiants représente le même individu. Du à la nature distribuée du Web, il peut exister des identifiants pour le même individu. L'utilisation de URL standard ne résout pas le problème. En effet, un individu peut avoir plusieurs URL, exemple une personne qui possède une page Web pour le travail et une autre personnelle, de même pour les emails.

Motivation : Interopérabilité d'ontologies, portail du web.

Exemple

Si une ontologie comporte des informations comme : « les publications académiques sont écrites par un ou plusieurs auteurs, qui sont des personnes identifiées par un nom, un prénom et une affiliation à des organismes ». Ceci permet d'identifier une personne. C'est ce genre d'inférence qu'une ontologie peut permettre.

10. ***Supposition de noms uniques locaux :*** En général, le langage doit construire des suppositions (hypothèses) de noms uniques, des identifiants qui ne sont pas associés à des objets différents. Cependant, il y a différentes applications où cette supposition de nom unique peut être utile. L'utilisateur doit

posséder la possibilité de spécifier que tous les noms sont définis dans un espace de noms particuliers ou document référent à des objets distincts.

Motivations : L'interopérabilité des ontologies.

11. ***Attachement des informations aux déclarations :*** Le langage doit offrir un moyen permettant aux déclarations d'être étiquetées avec des informations additionnelles comme la source, l'estampillage de temps, le degré de confidentialité, etc..., en définissant un ensemble de propriétés standard qui peuvent être utilisées comme tel. Il doit offrir à l'utilisateur, les mécanismes généraux pour attacher de telles informations.

Motivations : But partagé par tous les cas d'utilisation d'ontologies.

12. ***Classes comme instances :*** Le langage doit être en mesure de traiter les classes comme les instances. Cela parce que les mêmes concepts peuvent être vue comme classe ou comme individu, dépendant des perspectives de l'utilisateur.

Exemple : En ontologie biologique, la classe 'oiseaux' peut avoir des animaux individuels comme des instances. Cependant, 'oiseaux' peut être lui-même une instance de 'oiseaux-volants'. Noter que 'oiseaux' n'est pas une sous classe de 'oiseaux - volants', parce qu'il faut dire que toute instance de oiseau (un animal) est une instance de 'oiseaux- volants'. **Motivations :** Cas d'utilisation des ressources multimédias.

13. ***Types de données complexes :*** Le langage doit permettre la définition de types de données complexes ou structurées utilisées pour enregistrer les dates, n-uplets, les adresses, etc...

Motivations : Cas d'utilisation de calcul omniscient

14. ***Contraintes de cardinalités :*** Le langage doit permettre la restriction sur les cardinalités dans les propriétés. Ces restrictions englobent des bornes inférieures et supérieures de l'objet.

Motivations : Buts partagés par tous les objectifs d'un langage opérationnel et le cas d'utilisation de la documentation de développement.

15. ***Supporter le modèle de caractère*** : Le langage doit supporter l'utilisation de jeux de caractères multilingues.

Motivation : Internationalisation des ontologies.

16. ***Supporter le caractère unique de chaîne unicodes :*** Dans quelque codage de caractères, exemple : codage basé sur l'Unicode, il est nécessaire de disposer de moyen pour exprimer l'égalité de deux séquences différentes de caractères.

Exemple Un utilisateur déclarant une forme Pré-composée (juste un caractère 'ç') et un autre utilisant une forme décomposée (un caractère 'c' suivi d'une cédille), ne peuvent reconnaître les deux (02) caractères que s'ils ont choisi la même normalisation de code.

17. ***Référencement univoque des termes par URI :*** Deux termes d'ontologies différentes doivent avoir des identifiants absolus distincts. Il doit être possible d'identifier sans ambiguïté, un terme d'une ontologie par une référence URI.

III. Les Langages pour le Web Sémantique

La pyramide des langages proposée par W3C, représente un ensemble de langage dont seulement les couches basses sont stables par rapport à la normalisation et aux spécifications des langages (Figure 1). Cette organisation en pyramide donne en résultats deux (02) bénéfices [Lau03]:

- Permettre une approche graduelle dans les processus de standardisation et d'acceptation par les utilisateurs.

- Disposer d'un langage au bon niveau de complexité, celle-ci étant fonction de l'application à réaliser.

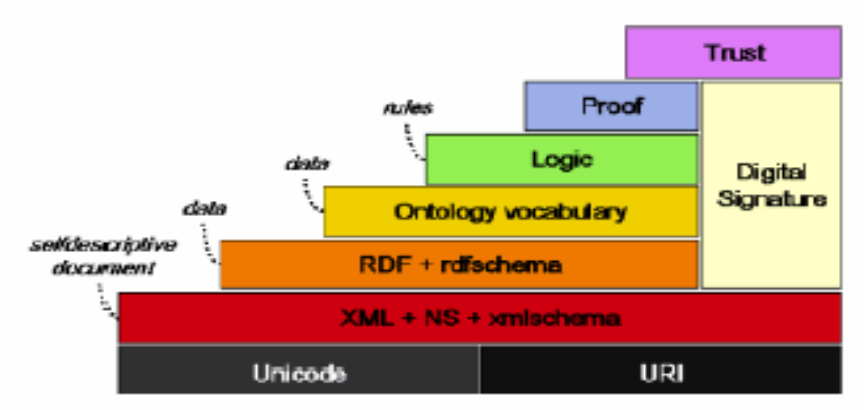

Figure 1. Les Couches du Web sémantique [Ber01]

Un aspect central de l'infrastructure est sa capacité d'identification et de localisation des diverses ressources. Elle repose sur la notion d'URI (Uniforme Resource Identifier) qui permet d'attribuer un identifiant unique à chaque ensemble de ressources accessibles ou non via le web. Cette notion connaît aujourd'hui de nombreuses extensions, en cour de standardisation, à d'autres entités que les URLs. Elle est à la base même des langages du W3C.

Une autre caractéristique de tous ces langages est d'être systématiquement exprimables et échangeables dans une syntaxe XML. Ceci permet de bénéficier de l'ensemble des technologies développées autours d'XML : XML Schéma, outils d'exploitation des ressources XML (Bibliothèques JAVA,etc.), bases de données gérant des fichiers XML, même si le langage de requêtes spécifiques est nécessaire pour les langages construits sur XML comme RDF.

III.1. Le Langage XML

XML est aujourd'hui un langage qui est au cœur d'une réflexion sur le web sémantique. On peut expliquer l'engouement pour cette technologie par l'obsolescence du langage le plus répondu sur Internet, le HTML. En effet, le langage HTML est figé et restrictif, c'est–à- dire qu'il n'autorise pas l'utilisateur à

définir ses propres balises. D'autre part, c'est un langage porté sur la présentation et non sur la description des données. C'est pourquoi le langage HTML est inefficace pour donner une dimension informationnelle aux données (i.e. pour exprimer des métadonnées).

XML, acronyme de « eXtensible Markup Language », peut être traduit par Langage Extensible de balisage. Il s'agit d'un langage de description des données, ou plutôt d'un métalangage permettant de décrire le contenu d'un document en dehors de sa présentation. Cette dernière va être définie dans une feuille de style (rattachée au document XML).

XML va permettre de structurer l'information contenue dans un document grâce à des balises [Mic99]. Et contrairement à HTML, XML offre la possibilité de créer ses propres balises. La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données.

Il est important de souligner qu'un document XML ne contient que des données. C'est pour cela que l'on peut considérer un document XML comme une base de données à part entière.

a. Avantages d'XML

- Très structuré : Stricte au niveau de la syntaxe et n'accepte pas d'erreurs

- Portable : La séparation des données de la présentation résulte à une compatibilité 100% avec les navigateurs.

- Très adapté à l'échange de données et de documents.

- Libre de droits, indépendant des plates-formes, correctement pris en charge et international.

b. Limites d' XML

Le langage XML, malgré tous ses points forts, souffre d'une lacune très importante c'est le manque de sémantique et c'est ce qu'on veut donner à l'information.

Soit l'exemple suivant, si je déclare qu'une personne qui porte le nom 'ADAM' est âgée de '13 ans', dont la pointure de pied est '9.5 pouce'. Les noms des balises qu'ils soient 'NAME' ou 'CCC', 'AGE' ou 'DDD', 'SIZE' ou 'EEE' désignent exactement les mêmes informations.

Donc, le code 1 et le code 2 sont équivalents du fait qu'on définit une structure pas un sens.

Code 1 :

```
< ?XML version = ''1.0'' encoding 'ISO-8859-1'' ?>
<Population>
    <Person id_person = ''ADAM''>
            <NAME> Adam</NAME>
            <AGE>13</AGE>
            <SHOESIZE> 9.5 Pouce <SHOESIZE>
```

```
    </Person>
</Population>
```

Code 2 :

```
< ?XML version = ‘’1.0’’ encoding ‘ISO-8859-1’’ ?>
<AAA>
    <BBB id_person = ‘’ADAM’’>
            <CCC> Adam</CCC>
            <DDD>13</DDD>
            <EEE>9.5 Pouce <EEE>
    </BBB>
</AAA>
```

On peut conclure que XML est un langage descriptif qui se contente de décrire les données. Grâce à sa flexibilité et à la possibilité de créer ses propres balises, on peut isoler toutes les informations élémentaires, mais on ne donne pas réellement de sens aux balises que l'on définit. donc, XML a une capacité limitée pour exprimer le sens des objets et les relations entre elles, sous forme d'un schéma ou d'une ontologie par exemple, mais ne définit pas de standard pour exprimer les métadonnées au moment où le Web sémantique doit permettre de donner du sens aux balises.

Par exemple

<livre> : Une balise de cette forme n'offre rien pour dire qu'un livre est un type de document, même chose pour dire que les balises <title>,<Author> informant que les documents ont un auteur, qui est une personne et un titre qui est un mot.

III.2. RDF Resource Description Framework

RDF [RDF04] est un modèle conceptuel (métalangage) normalisé par le W3C. Il est particulièrement ***destiné à la représentation de métadonnées*** pour une ressource web, comme le titre, la date de modification de la page Web, droit d'auteur et information sur la licence concernant un document Web. Cependant, en généralisant le concept de 'ressource web', RDF peut être aussi utilisé pour représenter des informations concernant des entités identifiables sur le web, ou ne peuvent être directement retrouvées à travers le Web, comme les livres d'une bibliothèques, les classes de cours d'une écoles, …etc.

Exemple : les documents incluant des informations concernant des articles disponibles d'un commerce en ligne (spécifications, prix, et disponibilité) ou description des préférences d'un utilisateur de Web.

RDF est plus adapté aux situations où l'information doit être traitée par diverses applications que d'être simplement affichée pour les utilisateurs. RDF offre un cadre commun pour exprimer des informations

échangeables entre les applications sans perte de sens [Hay04]. La possibilité d'échanger des informations entre les applications veut dire que l'information peut être disponible pour d'autres applications que celle productrice (originaire) de l'information.

RDF est basée sur l'idée d'identifier les choses en utilisant des identificateurs Web (appelé Uniform Resource Identifiers, ou URIs) et décrivant les ressources en terme de propriétés simples et de valeurs de propriété. Cela permet à RDF de représenter des déclarations simples concernant des ressources comme des noeuds et des arcs d'un graphe représentant des ressources, leurs propriétés et les valeurs associées.

Cependant, on peut dire que RDF offre déjà les possibilités suivantes [RDF04] :

- Une interopérabilité entre les applications qui échangent des informations que les machines peuvent comprendre.

- Des facilités pour permettre le traitement automatique des ressources Web.

- Les métadonnées de RDF peuvent être utilisées dans une variété de champs d'application.

Par exemple

1. Dans ***la recherche de ressources,*** offrir un meilleur mécanisme de recherche dans les catalogues pour décrire les contenus et les relations entre ces derniers.

2. Dans un ***site particulier***, page ou bibliothèque numérique par des agents intelligents pour faciliter le partage de connaissance.

3. Dans la ***description de collections*** de pages que représente un unique document logique, ou pour décrire les droits de propriété intellectuelle des pages web et beaucoup d'autres.

III.2.1. Le modèle de données élémentaire de RDF

Le modèle de données RDF est dédié à la description des ressources du web. Dans ses principes de base, RDF utilise trois types d'objets : une « ressource » (resource) décrite par des « propriétés » (properties) ; l'association d'une ressource à une propriété par une valeur de propriété est une « déclaration » (statement).

- ***Ressources.*** Toute chose décrite par RDF est appelée ressource. Une ressource peut être une page web entière, une partie d'une page web (un élément HTML ou XML spécifique à l'intérieur du document source), une collection de pages (un site Web complet), un objet qui n'est pas directement accessible via le net (un livre imprimé). Les ressources sont toujours normées par des URIs avec des identifiants optionnels.

- ***Propriétés :*** Une propriété est un aspect, une caractéristique, un attribut, ou une relation spécifique utilisée pour décrire une ressource. Chaque propriété possède une signification spécifique, définit ses

valeurs permises, les types de ressources qu'elle peut décrire, ainsi que les relations qu'elle entretient avec les autres propriétés.

- ***Valeur :*** Peut être soit une simple chaîne de caractères, soit une ressource.

Une *'Déclarations':* C'est un triplet <ressource, propriété, valeur> où la valeur est soit une ressource, soit un littéral.

Soit l'exemple suivant : Une personne nommé 'Mohamed' a crée une page Web particulière. La façon directe de faire cette déclaration en langage naturel est sous la forme d'une simple déclaration, comme :

<La page web 'http://www.example.org/index.html' a 'un créateur' dont la valeur est 'Mohamed'>.

Les parties de cette déclaration sont utilisées dans le but de décrire les propriétés de quelques choses :

- La ressource que la déclaration décrit (la page web, dans ce cas).
- Une propriété spécifique du sujet décrit dans la déclaration (créateur).
- L'information que la déclaration apporte, c'est la valeur de la propriété (qui est le créateur).

Dans cette déclaration, URL (Uniform Resource Locator) 'http://www.example.org/index.html' de la page Web est utilisé pour identifier cette page. En addition, le mot 'Créateur' est utilisé pour identifier la propriété, et le mot « Mohamed » pour identifier la chose (la personne) c'est la valeur de cette propriété.

Des informations additionnelles sur cette page peuvent être exprimées comme sa date de création ou sa langue comme suit :

'http://www.example.org/index.html a une date-création dont la valeur est 16 Août 1999.

http://www.example.org/index.html a un langage dont la valeur est Arabe.

RDF utilise une terminologie particulière pour préciser des parties variées d'une déclaration <Sujet, Prédicat, Objet> :

- Le sujet de cette déclaration est la page Web http://www.example.org/index.html
- La propriété ou la caractéristique du sujet, spécifiée par la déclaration est : créateur, date-création, ou le langage, est appelé le Prédicat.
- La valeur de cette propriété est appelée l'Objet : John Smith, anglais, 16 Août 1999.

Cependant, le langage naturel est suffisant pour la communication entre des humains, au moment où le but est de rendre ces types de déclaration traitable par une machine. Pour cela, il est nécessaire d'avoir :

- Un système d'identificateurs traitable par les machines pour identifier le sujet, le prédicat ou l'objet dans une déclaration sans aucune possibilité de confusion avec un identificateur qui paraît similaire et qui peut être utilisé par quelqu'un d'autre sur le Web (associer des URLs aux parties d'une déclaration comme il est claire dans la figure 3).

- Un langage traitable par une machine pour représenter ces déclarations et les échanger entre machines.

III.2.2. Représentation schématique des déclarations RDF :

La schématisation des déclarations RDF respecte les règles suivantes :

- Une ressource est représentée par en forme d'ellipse,
- Un prédicat (propriété) est représenté par un arc,
- Un objet est représenté par un rectangle.

Voici les graphes correspondants aux déclarations vues précédemment :

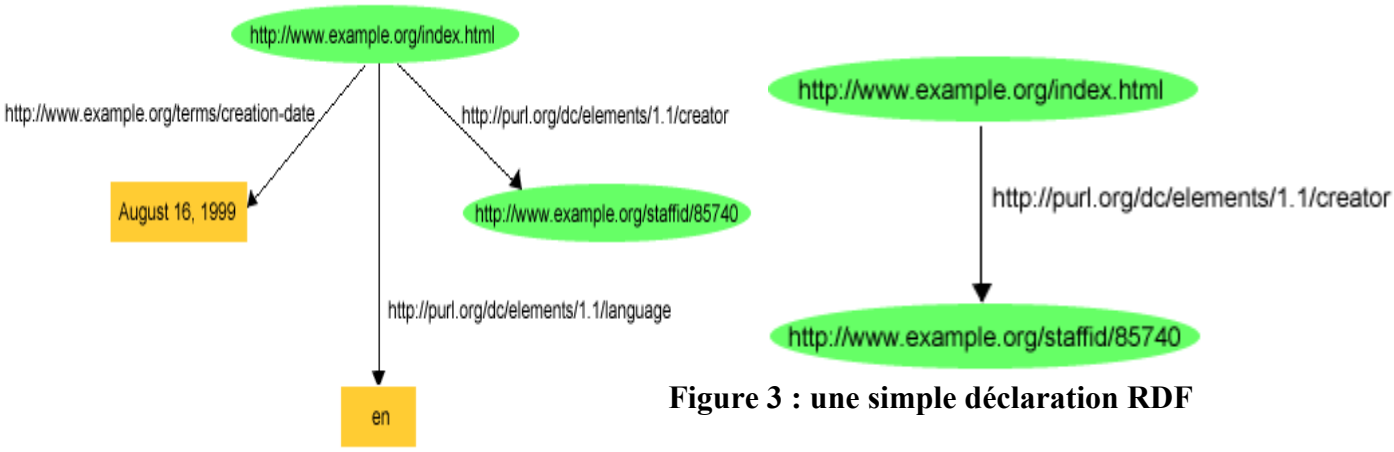

Figure 2 : différentes déclarations concernant la même ressource

Figure 3 : une simple déclaration RDF

Si on veut par exemple dessiner le graphe correspondant à la déclaration RDF suivant « Ali dont l'identifiant est 85740 a une adresse « Rue du Hoggar, Alger, code postal 16035 »

Le graphe correspondant sera

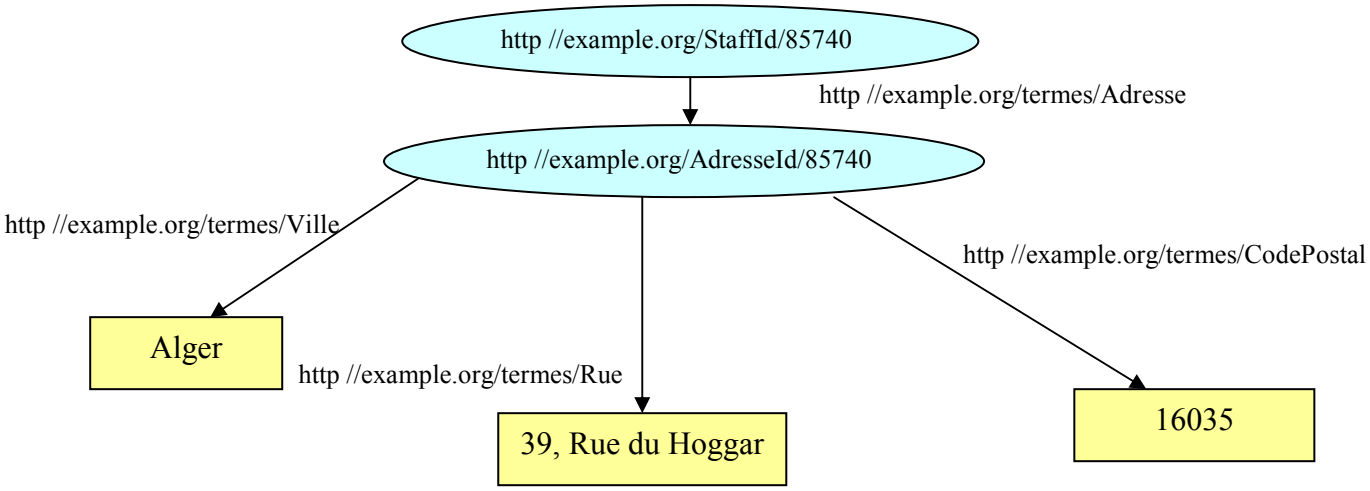

Figure 4. Déclaration avec une ressource intermédiaire identifiée

Dans cette déclaration, on a utilisé une ressource intermédiaire identifiée par « exAdresseId :85740 », qui a permis de représenter un concept composé comme l'adresse. Cette ressource une fois les composants identifiés, ne servira plus à grand chose. Voici la déclaration correspondance sous forme de triplets :

exstaffId : 85740	extermes :Adresse	exAdresseId : 85740.
exAdresseId : 85740	extermes : Rue	« 39, Rue du Hoggar ».
exAdresseId : 85740	extermes : Ville	« Alger »
exAdresseId : 85740	extermes : CodePostal	« 16035 ».

Il est possible dans RDF d'utiliser un nœud blanc (sans URI) pour regrouper les déclarations appartenantes concernant la même ressource, Adresse dans ce cas, comme suit :

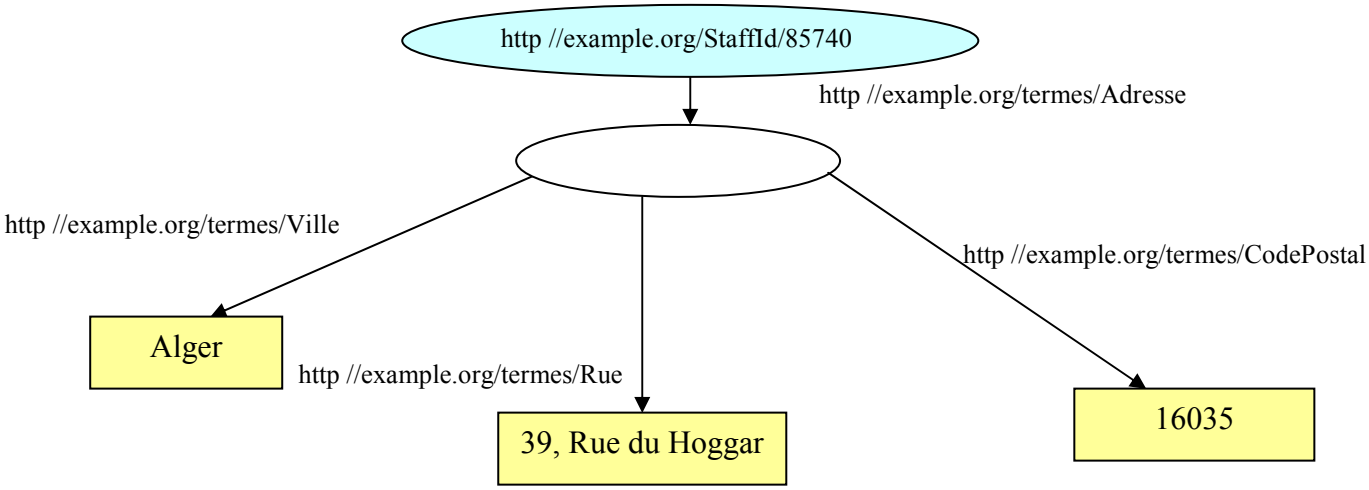

Figure 5. Déclaration avec un nœud blanc

III.2.3. La syntaxe élémentaire de RDF

Pour enregistrer et échanger les graphes, RDF offre une syntaxe basée sur XML appelée RDF/ XML. Il est à noter que RDF/ XML contient des URI, les propriété et leurs valeurs.

Comme pour HTML, RDF/ XML est traitable par les machines. En utilisant les URI, il peut assembler des pièces d'information à travers le web. Cependant, différent de l'hypertexte conventionnel, les URIs de RDF peuvent faire référence à n'importe quel objet (chose) sur le web ou dans le mode réel qui ne peuvent être directement accessible à travers le web comme les voitures, les affaires, les gens, …etc. En addition, les propriétés RDF elles-mêmes possèdent des URI, pour identifier précisément les rapports qui existent entre les objets (items) liés.

La représentation du modèle de données de la figure 6 en RDF/XML est la suivante :

http://www.example.org/index.html a une creation-date dont la valeur est 16 Août 1999

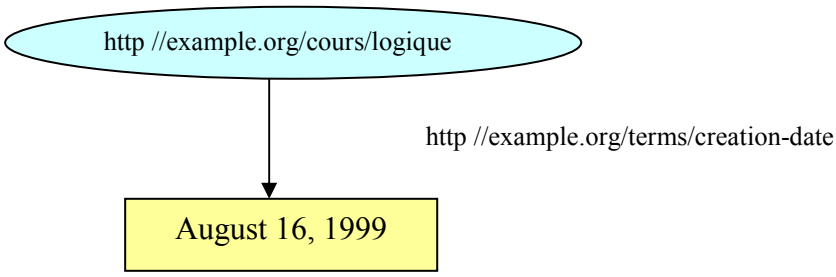

Figure 6. Exemple d'un schéma d'une déclaration RDF

Avec la représentation du triplet: <example.org/index.html, example.org/terms/creation-date, "August 16, 1999">

Et en RDF/ XML

```
1. <? Xml version =”1.0”?>
2. <rdf: RDF xmlns:rdf=’http//www.w3.org/1999/02/22-rdf-syntaxe-ns#”
3.                         Xmlns:exterms=’http://www.example.org/terms/“>
4. <rdf:Description about=’’http://www.example.org/index.html”>
5.                          <exterms :creation-date> 16 Aout 1999</exterms :creation-date>
6. </rdf:Description>
7. </rdf:RDF>
```

Chaque élément RDF est un simple emballage. Celui-ci marque les frontières au sein d'un document XML entre lesquelles le contenu est explicitement défini à l'intérieur d'une instance de modèle de donnée RDF.

Dans cet exemple, on peut retrouver une définition d'un espace de nom qui a été déclaré grâce à la notation 'xmlns' qui permet de référencer chaque définition de la syntaxe de RDF. Chaque instruction commençant par « RDF » fera référence à cet espace de nom. Le second est appelé 'exterms' permet de rendre disponible, par exemple, un ensemble de termes.

Concernant <rdf:Description about=’’http://www.example.org/index.html”>, 'Description' peut être vue simplement comme un lieu pour contenir l'identification de la ressource en cours de description. Typiquement, il y aura une ou plusieurs déclarations faites pour une ressource, 'Description' fournit un moyen de donner une seule fois le nom de la ressource pour plusieurs déclarations.

Lorsque l'attribut 'about' est spécifié, les déclarations dans 'Description' se réfèrent à la ressource dont l'identifiant est déterminé par 'about'.

Si la portion du code suivant est ajoutée

<exterms : Créateur rdf :resource=''<tt>http://www.w3.org/staffId/85740</tt>''/>

Si 'exterms' est l'espace de noms déjà défini et 'créateur' fait partie du vocabulaire de cet espace de noms, alors cela signifie que la personne dont l'identificateur est 85740 est le créateur de la ressource 'http://www.example.org/index.html'.

III.2.4. Autres possibilités de RDF

RDF offre d'autres possibilités, comme incorporé des types et des propriétés pour représenter des groupes de ressources et des déclarations RDF ainsi que des possibilités d'insérer des fragments écrits en XML comme des valeurs de propriétés et les types de données XML. Ces possibilités additionnelles sont décrites dans ce qui suit :

a. Les conteneurs

C'est souvent le besoin de décrire un *Groupe* d'objets, comme par exemple, les auteurs d'un livre ou les étudiants d'un cours ou encore les composants d'un logiciel que la notion d'ensemble a été introduite. RDF offre une variété de types prédéfinis et des propriétés qui peuvent être utilisées pour décrire de tels groupes.

Un conteneur est une ressource qui contient des objets appelés '*Membres*'. Les membres d'un conteneur peuvent être des ressources ou des littéraux. RDF définie trois sous types:

- rdf :Bag
- rdf :Seq.
- Rdf :Alt.

'Bag ': représente un groupe de ressources ou littéraux, qui peut contenir des membres dupliqués, où il n'y a pas de signification à l'ordre attribué à ces derniers.

Séquence ou 'Seq' représente un groupe de ressources ou littéraux, pouvant aussi contenir des duplications. L'ordre attribué aux membres est signifiant.

Alternative ou 'Alt' représente un groupe de ressources ou littéraux qui sont alternatives (typiquement pour une valeur unique d'une propriété). Par exemple, une 'Alt' peut être utilisée pour décrire des translations alternatives de langages pour le titre d'un livre, ou pour décrire une liste de sites Internet alternative dans lesquels la ressource peut être retrouvée. Une application utilisant une propriété dont la valeur est un conteneur 'Alt' peut choisir n'importe quel membre du groupe comme approprié.

Exemple 1

Pour représenter la déclaration: Dans le cours de 'logique', les étudiants sont Ali, Mohamed, Lila et Réda,

Le cours doit être décrit en donnant sa propriété s:étudiants (dans un vocabulaire approprié). La valeur de cette propriété est un conteneur de type rdf :bag (représentant le groupe des étudiants). Les étudiants doivent être identifiés comme des membres de ce groupe, comme il est clairement représenté dans ce graphe.

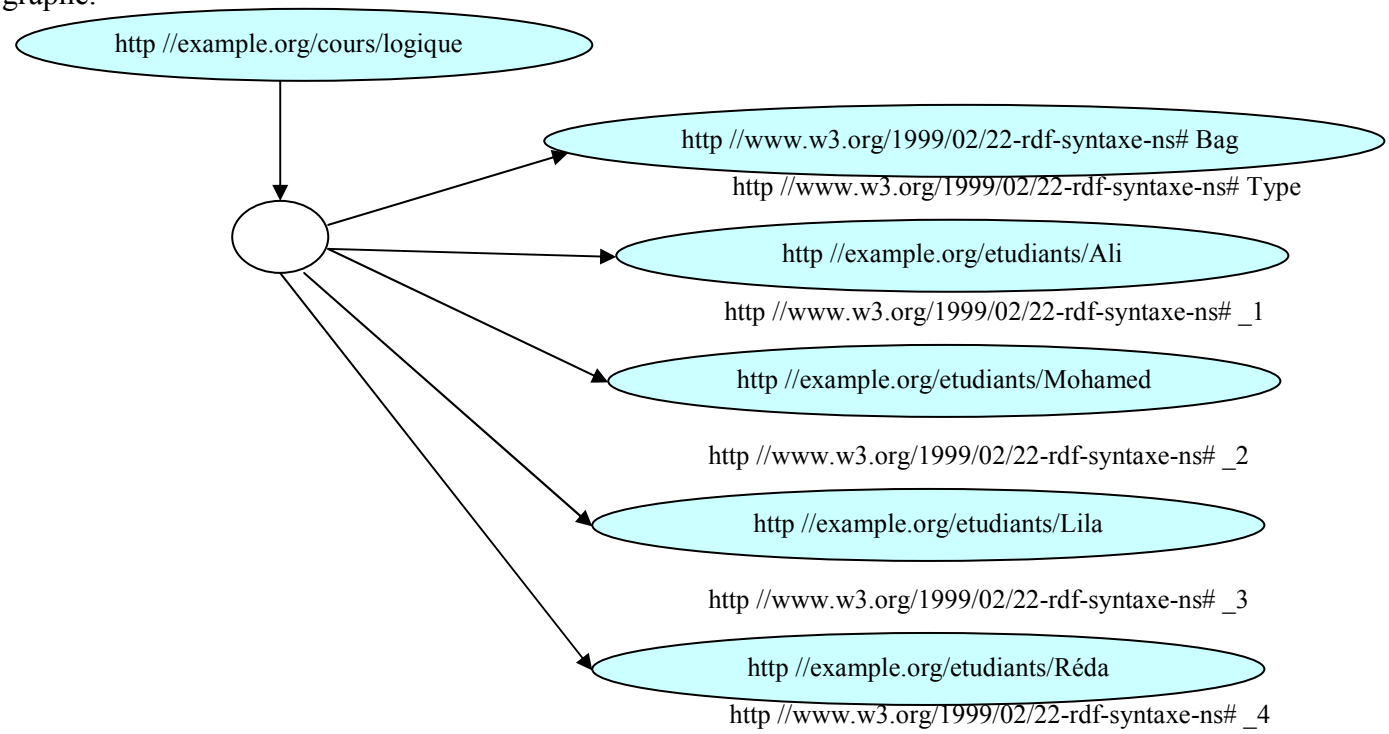

Figure 7. Une simple description d'un conteneur de type Bag

Depuis que la valeur de la propriété de s : étudiants dans cet exemple est décrite à l'aide d'un 'Bag' alors il n'y a aucune signification attribuée à l'ordre.

RDF/ XML offre une syntaxe assez spéciale et des abréviations pour le rendre simple dans la description des conteneurs. Le graphe précédent peut être écrit comme suit :

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:s="http://example.org/students/vocab#">
  <rdf:Description rdf:about="http://example.org/cours/logique">
    <s:étudiants>        <rdf:Bag>
                          <rdf:li rdf:resource="http://example.org/students/Ali"/>
                          <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
                          <rdf:li rdf:resource="http://example.org/students/Lila"/>
                          <rdf:li rdf:resource="http://example.org/students/Réda"/>
                        </rdf:Bag>
```

```
    </s:étudiants>
  </rdf:Description>
</rdf:RDF>
```

La valeur de la propriété ‘étudiants’ est formulée comme un ‘Bag’. La forme généralisée ‘rdf :li’ permet de définir les différentes ressources du conteneur. Les conteneurs peuvent avoir des identifiants.

Exemple2

Pour illustrer un conteneur Alt, représentons la phrase ‘le code source pour X11 peut être retrouvé dans ftp.exemple.org.ftp1 ou ftp1.exemple.org, ou ftp2.example.org ’. Elle peut être exprimée par le graphe RDF dans ce qui suit :

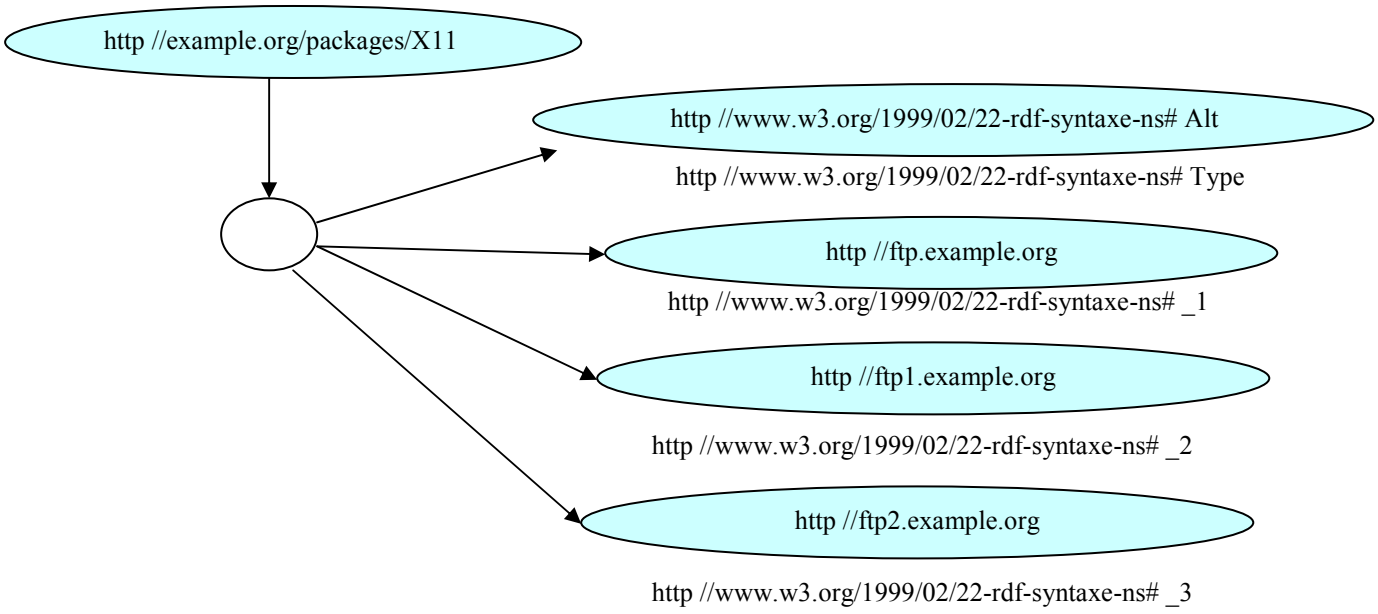

Figure 8. Une simple description d’un conteneur de type Alt

Et dont l’écriture en RDF/XML est la suivante :

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/packages/vocab#">
  <rdf:Description rdf:about="http://example.org/packages/X11">
    <s:DistributionSite>
      <rdf:Alt>          <rdf:li rdf:resource="ftp://ftp.example.org"/>
                         <rdf:li rdf:resource="ftp://ftp1.example.org"/>
                         <rdf:li rdf:resource="ftp://ftp2.example.org"/>
      </rdf:Alt>
    </s:DistributionSite>
  </rdf:Description>
</rdf:RDF>
```

Il n'existe pas de méthode pour fermer un conteneur, ou pour s'assurer qu'il y a tous les membres du conteneur. Le conteneur exprime seulement qu'un certain nombre de ressources fait partie de l'ensemble, mais aucune façon pour exprimer qu'il n'existe pas d'autre. Donc, quand un graphe peut décrire certains de ces membres, il n'existe pas de façon d'exclure les possibilités qu'un autre graphe décrive des membres additionnels. RDF offre des supports pour décrire des graphes contenant seulement des membres spécifiques. Pour cela on définit les collections.

b. Collection RDF

Une collection RDF est un groupe d'objets représenté comme une structure de liste dans un graphe RDF. Cette liste est construite en utilisant un vocabulaire de collection prédéfinie consistant en un type prédéfini '*rdf : List'*, une propriété prédéfinie '*rdf :First'* pour la première ressource et '*rdf :rest'* pour le reste de la liste en excluant la tête ainsi que la ressource prédéfinie '*rdf :Nil'* pour fermer le groupe.

Pour illustrer cela, soit la phrase 'les étudiants dans le cours « logique » sont seulement 'Ali, Mohamed et Lila'. Cette phrase va être représentée en utilisant le graphe comme suit :

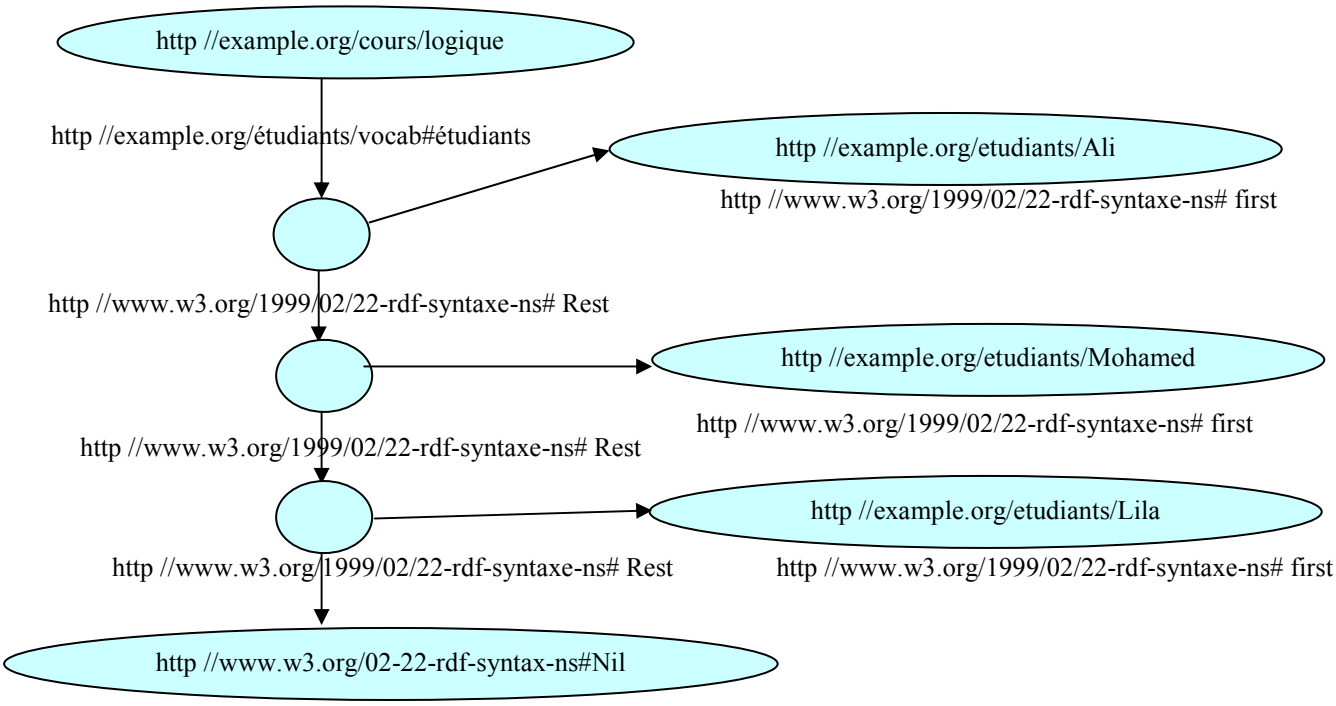

Figure 9. Une collection RDF

Dans ce graphe, chaque membre de la collection, comme 'S :Ali' est un objet de la propriété 'rdf :First' dont le sujet est une ressource qui représente une liste. Ce membre est lié au reste de la liste par la propriété 'rdf :rest'. La ressource 'rdf :Nil' représente une liste vide.

RDF/XML offre une notation spéciale pour faciliter la description de collection en utilisant des graphes de cette forme (voir Figure 8). Dans RDF/ XML, une collection peut être décrite par un élément propriété qui est l'attribut 'rdf :parseType' pour indiquer que les éléments d'une collection peuvent d'être interprétés de façon spéciale.

La portion du code approprié à ce graphe est la suivante :

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://example.org/étudiant/vocab#">
  <rdf:Description rdf:about="http://example.org/cours/logique">
    <s:étudient rdf:parseType="Collection">
        <rdf:Description rdf:about="http://example.org/étudients/Ali"/>
        <rdf:Description rdf:about="http://example.org/étudients/Mohamed"/>
        <rdf:Description rdf:about="http://example.org/étudients/Lila"/>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

c. La Réification

Les applications ont besoin quelques fois de décrire des déclarations RDF en utilisant d'autres *d'ordre supérieur*.

RDF offre un vocabulaire incorporé dédié pour décrire des déclarations RDF. La description d'une déclaration en utilisant ce vocabulaire est appelée 'La Réification d'une déclaration'. Le vocabulaire d'une réification RDF consiste en un type 'rdf :statement', et les propriétés 'rdf :subject', 'rdf :predicate' et 'rdf :object', ont les significations suivantes :

- **Predicate :** La propriété '*predicate'* identifie la propriété originale dans la déclaration modélisée. La valeur de la propriété '*predicate'* est une ressource qui représente la propriété spécifique dans la déclaration originale.

- **Subject** : La propriété '*subject'* identifie la ressource décrite par la déclaration modélisée. La valeur de la propriété *'subject'* est la ressource pour laquelle la déclaration originale a été faite.

- **Object :** La propriété '*object'* identifie la valeur de la propriété dans la déclaration modélisée. La valeur de la propriété '*object'* est l'objet dans la déclaration originale.

- **Type :** La valeur de la propriété *type* décrit le type de la nouvelle ressource. Toutes les déclarations réifiées sont des exemples de '*RDF :Statement'*. Elles ont une propriété type dont l'objet est '*RDF :Statement'*. La propriété type est aussi plus généralement utilisée pour déclarer le type de n'importe quelle ressource.

Exemple

Pour déclarer que la ressource identifiée par ‘exproductions :triple12345’ est une déclaration RDF, et que le sujet de cette déclaration réfère à la ressource identifiée par ‘exproducts :item10245’, et que le prédicat de cette déclaration est identifiée grâce à la ressource ‘exterms :Weight’, et l’objet est la valeur décimale identifiée par le littéral typé ‘’2.4’’ ^^xsd ; decimal. Dans ce qui suit, on déclare cet exemple par son code RDF ainsi que son graphe associé.

```
<?xml version="1.0"?>
    <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:dc="http://purl.org/dc/elements/1.1/"
             xmlns:exterms="http://www.example.com/terms/"
             xml:base="http://www.example.com/2002/04/products">
      <rdf:Description rdf:ID="item10245">
        <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
      </rdf:Description>
  <rdf:Statement rdf:about="#triple12345">
        <rdf:subject rdf:resource="http://www.example.com/2002/04/products#item10245"/>
        <rdf:predicate rdf:resource="http://www.example.com/terms/weight"/>
        <rdf:object rdf:datatype="&xsd;decimal">2.4</rdf:object>
        <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
 </rdf:Statement>
</rdf:RDF>
```

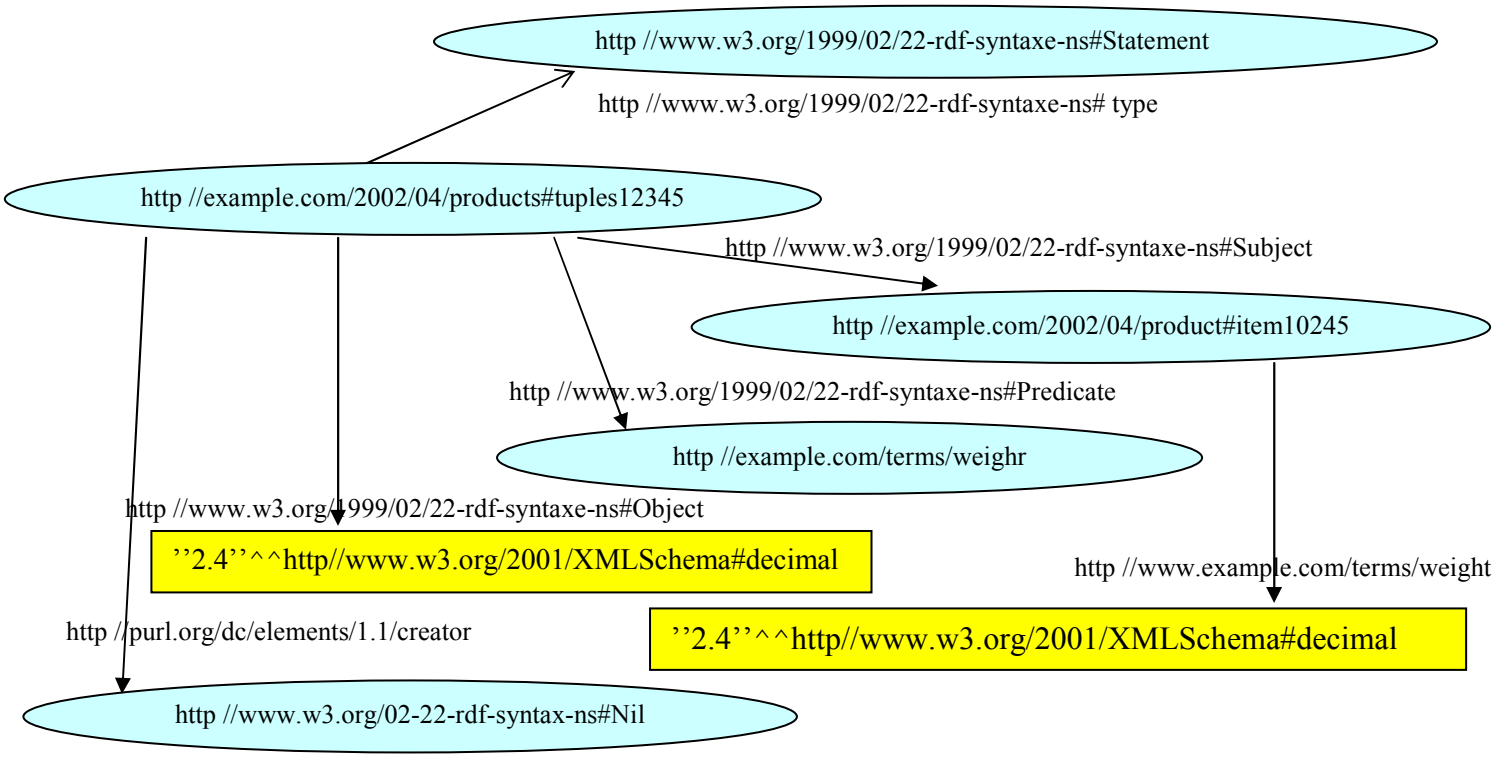

Figure 10. Une déclaration, sa réification et ses attributs

III.2.5. Les insuffisances de RDF

Le langage RDF permet de définir formellement des métadonnées pour une large catégorie de données. A travers sa syntaxe écrite principalement en XML, les applications peuvent comprendre une grande partie de la traduction des données. Le point faible de ce langage est qu'on ne peut pas définir, des liens de spécialisation ni d'instanciation. C'est ce qui a poussé à son une extension connue sous le nom 'RDF Schéma – RDFS'.

Exemple [RDF 04]

Si on veut formuler en RDF la déclaration suivante : « un livre ou un article d'une magazine possède un titre ou un sujet ».

Il est nécessaire de définir une classe '*ex : Livre'* ou '*ex :MagazineArticle*' et les propriétés '*ex :Sujet*' ou '*ex :Titre*' pour la décrire, ce qui n'est pas permis dans RDF. RDF ne permet pas de déclarer qu'un magazine est un ensemble d'articles, et qu'un livre est un ensemble de chapitres. De plus, il ne permet pas de dire que '*Sujet'* et '*Titre'* désigne le même élément.

RDF lui-même n'offre aucun moyen permettant d'exprimer de telles définitions. Pour répondre aux besoins des différentes applications, son extension RDFS décrit ces types de classes et de propriétés tout en gardant le même vocabulaire que celui de RDF.

III.3. RDF Schema (RDFS)

RDFS a pour but d'étendre RDF en décrivant plus précisément les ressources utilisées pour étiqueter les graphes. Pour cela, il fournit un mécanisme permettant de spécifier les classes dont les ressources seront des instances, ainsi que les propriétés. RDFS s'écrit toujours à l'aide de triplets RDF, en définissant la sémantique de nouveaux mots-clés comme :

- <ex :Vehicule rdf :Type rdfs :Class> la ressource 'ex :Vehicule' a pour type 'rdfs :Class', et est donc une classes ;

- <sncf :TER8153 rdf :type ex :Vehicule> la ressource 'sncf :TER8153' est une instance de la classe 'ex :Vehicule', déjà défini.

- <sncf :Train rdfs :SubClassOf ex :Vehicule> la classe 'sncf :Train' est une sous-classe de 'ex :Vehicule'. Toutes les instances de 'sncf :Train' sont donc des instances de 'ex :Vehicule' ;

- <ex :Localisation rdf :type rdfs :Property> affirme que ex :localisation est une propriété (une ressource utilisable pour étiqueter les arcs).

- <ex :localisation rdfs :range ex :Ville> affirme que toute ressource utilisée comme extrémité d'un arc étiqueté par 'ex :localisation' sera une instance de la classe 'ex :Ville'.

RDFS ajoute à RDF les possibilités de définir des hiérarchies de classes et de propriétés dont l'applicabilité et le domaine de valeurs peuvent être contraintes à l'aide des attributs 'rdfs :domain' et 'rdfs :range'. A chaque domaine applicatif, on peut associer un schéma identifié par un préfixe particulier et correspondant à une URI. Les ressources instances sont ensuite décrites en utilisant le vocabulaire donné par les classes définies dans ce schéma. Les applications peuvent alors leur donner une interprétation opérationnelle.

III.3.1. Les classes

RDFS définit un certain nombre de ressources comme classes de base de son vocabulaire :

'rdf : Resource' toutes choses décrites par des expressions RDFS est appelée ressource, qui est instance de la classe 'rdf :Resource'.

'rdf :Property' est un sous ensemble des ressources RDF. On l'utilise pour étiqueter les arcs d'un graphe RDF.

'rdfs : Class' : L'étape de base dans tout type de description de processus est l'identification des objets à décrire. RDF Schéma se réfère à 'ces types d'objets' par la notion de « classe ».

Une classe RDFS correspond à un *concept générique*, à un *type* ou à une *catégorie.* On peut l'assimiler la notion de classe dans les langages de programmation orientés objet comme dans Java. Les classes RDF

peuvent décrire différents types d'objets comme les pages web, les personnes, les types de documents, les bases de données ainsi que les concepts abstraits.

III.3.2. Les Propriétés

En plus des possibilités de décrire des classes spécifiques d'objets, la communauté d'utilisateurs a aussi besoin de décrire des propriétés spécifiques qui caractérisent ces classes d'objets. Ce sont des instances de la classe **'rdf :Property'**. RDFS offre un mécanisme d'expression des relations entre les classes et leurs instances ou super-classe.

'rdf :Type' : associe à une classe un type précis, et indique que cette ressource possède toutes les caractéristiques qu'un membre de cette classe doit avoir. En respectant les règles d'héritage et d'instanciation, une ressource peut être instance de plusieurs super classe.

'rdfs :SubClassOf' : est une propriété transitive restreinte à la classe 'rdfs :Class'. Une classe peut être une sous-classe de plusieurs classes. Cependant, une classe ne peut jamais être déclarée comme étant une sous-classe d'elle-même.

'rdfs :subPropertyOf' : est une instance de 'rdf :Property'. Elle est employée pour spécifier qu'une propriété est une spécialisation d'une autre. Une propriété ne peut jamais être instance d'elle-même.

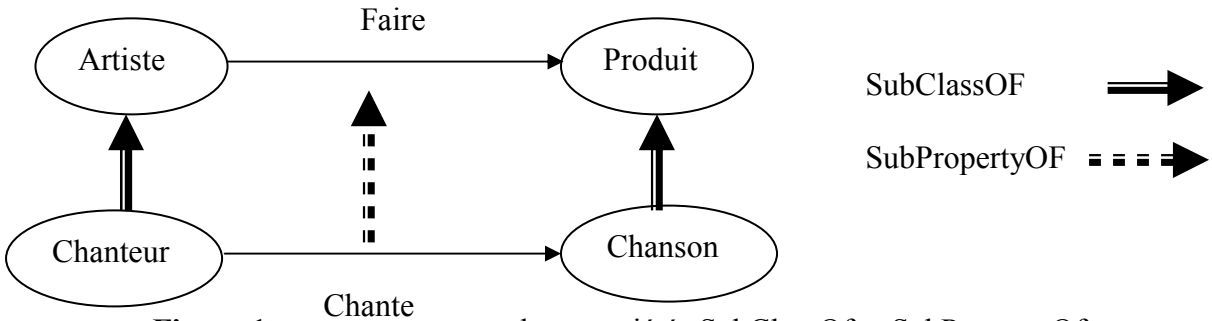

Figure 11. Schéma utilisant les propriétés SubClassOf et SubPropertyOf**.**

'rdfs :seeAlso' : c'est un moyen de fournir des informations additionnelles sur la ressource sujet. Elle peut être spécialisée en utilisant 'rdfs :subPropertyOf' afin d'indiquer la nature des informations que l'objet source a sur la ressource sujet.

'rdfs :isDefinedBy' : c'est une sous-propriété de 'rdfs :seeAlso'. Elle sert à indiquer la source qui a définit la ressource sujet.

III.3.3. Les Contraintes

RDFS fournit un mécanisme de contraintes :

'rdfs:ConstraintResource' : elle permet de définir une sous-classse de 'rdf :Resource' dont les instances sont les concepts de RDFS impliqués dans l'expression des contraintes. Le but de cette classe est de fournir un mécanisme de définitions de contraintes associées à un modèle RDF.

'rdfs :ConstraintProperty' : sous-classe de 'rdf :Property', dont les instances sont des propriétés utilisées pour spécifier des contraintes. Cette classe est une sous-classe de 'rdfs :constraintResource' et correspond au sous-ensemble de cette classe représentant les propriétés.

'rdfs :range' instance de 'ConstraintProperty'. Elle est utilisée pour indiquer à quelles classes les valeurs d'une propriété appartiennent. La valeur d'une propriété 'range' est toujours une classe. Les contraintes 'range' sont seulement appliquées aux propriétés.

'rdfs :domain' est une instance de 'ConstraintProperty' qui est utilisée pour indiquer aux classes sur quels membre une propriété peut être utilisée.

II.3.4. La documentation

'rdfs :comment' est utilisée pour ajouter des commentaires dans la description d'une ressource.

'rdfs :label' permet d'ajouter un nom à une ressource.

II.3.5. Autres possibilités de RDFS

'rdfs :Literal' correspond à l'ensemble appelé 'Literals' dans un modèle formel de RDF.

'rdfs :Container' : est la classe utilisée pour représenter les groupes de types Container.

'**rdfs :ContainerMembershipProperty'** : cette classe a pour membres les propriétés _1,_2,_3,…, utilisées pour indiquer les membres d'un conteneur.

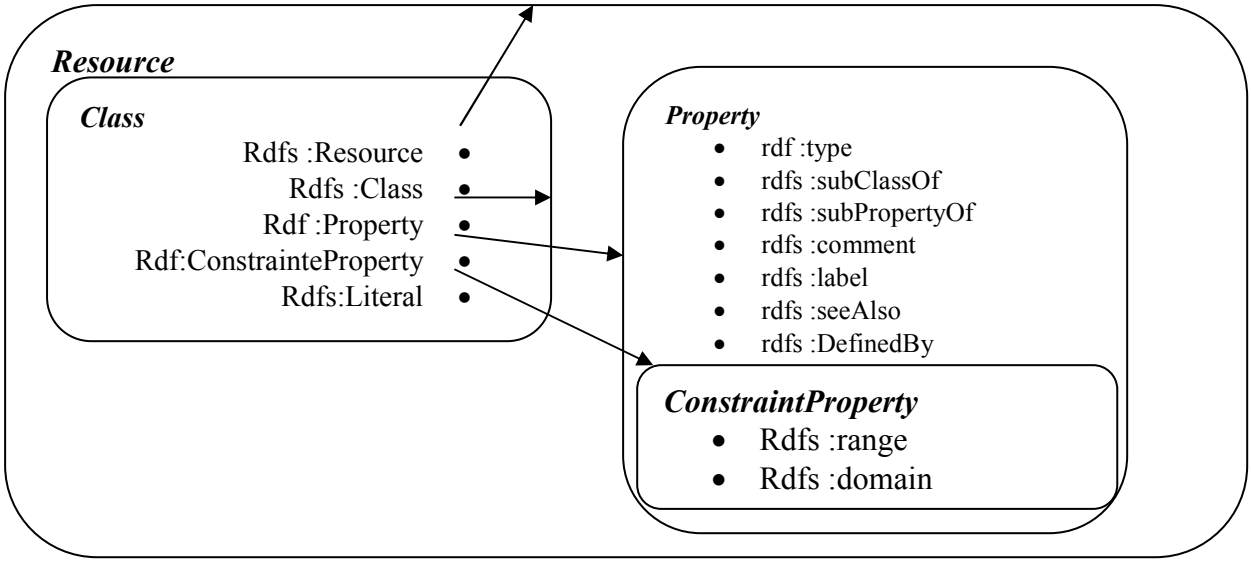

Figure 12. Les différents concepts du langage RDFS

Exemple d'utilisation de RDF- RDFS :

On veut définir une classe 'Etudiant', comme sous-classe de la classe 'Personne'. On déclare ensuite une propriété 'Matricule' qui s'applique sur les objets de la classe 'Etudiant' et prend ses valeurs dans la classe 'Etudiant-Matricule'. Elle est représentée en RDF- RDFS comme suit :

```
<rdfs:Calss rdf :ID=’Etudiant’>
        <rdfs:subClassOf rdf:resource=’#Personne’’/>
</rdfs:Class>
<rdf:Property rdf:ID=’’Matricule’’>
        <rdfs:domain rdf:resource=’’#Etudiant’’/>
        <rdfs:range rdf: resource=’’#Etudiant-Matricule’’/>
</rdf:Property>
```

Si on veut instancier un nouvel étudiant, on le fait comme suit :

```
<Etudiant id=‘1236’>
<rdfs :label> Matricule de l’étudiant </rdfs :label>
<rdfs :comment> les matricules des étudiants sont codés séquentiellement </rdfs :comment>
</Etudiant>
```

IV. Conclusion

La manipulation des ressources du Web par les machines requiert l'expression ou la description de ces ressources. Plusieurs langages sont donc définis. A cet effet, ils doivent permettre d'exprimer les données et métadonnées (RDF), et de disposer d'un modèle abstrait de ce qui est décrit grâce à l'expression d'ontologies (RDFS).

Le modèle de données de RDF, ainsi que sa représentation graphique sont très proches d'un des modèles de représentation de connaissances, les graphes conceptuels. Le chapitre suivant, sera consacré à présenter en détail ce formalisme afin de répertorier les points communs ainsi que les différences entre les deux modèles (RDF, graphe conceptuel) dans le but d'enrichir la description des ressources sur le web par les capacités sémantiques disponibles dans les graphes conceptuels.

Chapitre 3

Chapitre III

Les graphes conceptuels et les graphes Conceptuels flous

I. Introduction

Les graphes conceptuels sont un système de représentation de la sémantique du langage naturel. Ils sont conçus pour simplifier le passage du et au langage naturel. Ils sont basés sur une notation de graphes pour la logique développée en premier par le philosophe et logicien C.S. Peirce [Pei82].

Les premiers graphes de Peirce étaient simples. Mais, ils n'étaient pas suffisamment généraux pour représenter toute la logique [Sow91]. En 1883, il a développé sa notation linéaire, qui avec un minimum de modification, a été utilisée dans le calcul des prédicats moderne. En 1897, Peirce a proposé une extension de ces graphes pour représenter toute la logique du premier ordre, et par la suite, la logique modale. Son système, appelé 'graphes existentiels', constitue le fondement logique pour les « graphes conceptuels ».

Le terme 'graphe conceptuel' désignait à la fois un modèle de base, défini d'une manière assez précise, et appelé parfois 'graphes conceptuels simples', et des extensions, ou des idées d'extension, du modèle de base définies de manière plus ou moins formelles [Mug96].

Parmi les points positifs ou les avantages derrière l'utilisation des graphes conceptuels, on peut citer :

- Facilité de représentation graphique: « un bon dessin vaut mieux qu'un long discours ». Cette représentation basée sur l'objet, les relations entre les divers objets sont claires et visibles.

- Un modèle de type graphe étiqueté est intéressant d'un point de vue informatique pour diverses raisons. De nombreux algorithmes efficaces ont été développés pour résoudre divers problèmes sur les graphes.

- Un graphe conceptuel est 'plus facile à lire' que la formule logique qui lui est associée. Exemple : soit la phrase suivante : le chat est sur le tapis. La formule correspondante dans le calcul des prédicats : $(\exists x)(\exists y)(chat(x) \wedge tapis(y) \wedge sur(x, y))$ tandis que le graphe correspondant est : [Chat] →(Sur)→[Tapis]. Le graphe montre directement les connexions. Les variables x et y obligent l'œil à scanner la formule pour déterminer comment les parties de la formule sont reliées. De plus, la formule introduit deux conjonctions où ni la phrase en langage naturel ni le graphe n'en possèdent.

- D'un point de vu algorithmique : Les inférences se calculent par des algorithme de graphes, et les algorithmes polynomiaux déjà construits. De plus, la projection, qui est l'opération fondamentale, et qui correspond à l'implication logique, est facilement visualisable. Un utilisateur peut donc suivre visuellement un 'raisonnement'.

- Facilité d'extension des capacités descriptives du modèle des graphes conceptuels simples : extension du modèle de base.

II. Modèle des graphes conceptuels

Le modèle des graphes conceptuels se compose de deux (02) parties à savoir : le niveau terminologique et le niveau assertionnel :

II.1. le niveau terminologique

Le niveau de connaissance terminologique dans le modèle des graphes conceptuels est appelé 'le support'. Il contient le vocabulaire de base sur lequel vont être construits les graphes conceptuels. Ce vocabulaire est réparti en trois ensembles : les types de concepts, les types de relations, et les marqueurs. Chacun de ces ensembles est muni d'une relation d'ordre partiel, définissant la relation de spécialisation/ généralisation.

- L'ensemble des types de concepts T_c est un ensemble fini et encapsule les caractéristiques communes de plusieurs concepts. Cet ensemble est partiellement ordonné par la relation 'sorte de' et est muni d'un plus grand élément (sur_type de tout type) et d'un plus petit élément (sous_type de tout type) notés respectivement $\top$ et $\bot$. Ils sont respectivement appelés type universel et type absurde.

Les types de concepts représentent les entités, les états, les attributs, les évènements et les actions, etc… .

Exemple

Dans la figure 1, représente une hiérarchie de types de concepts. Les liens entre les concepts représentent les relations de spécialisation.

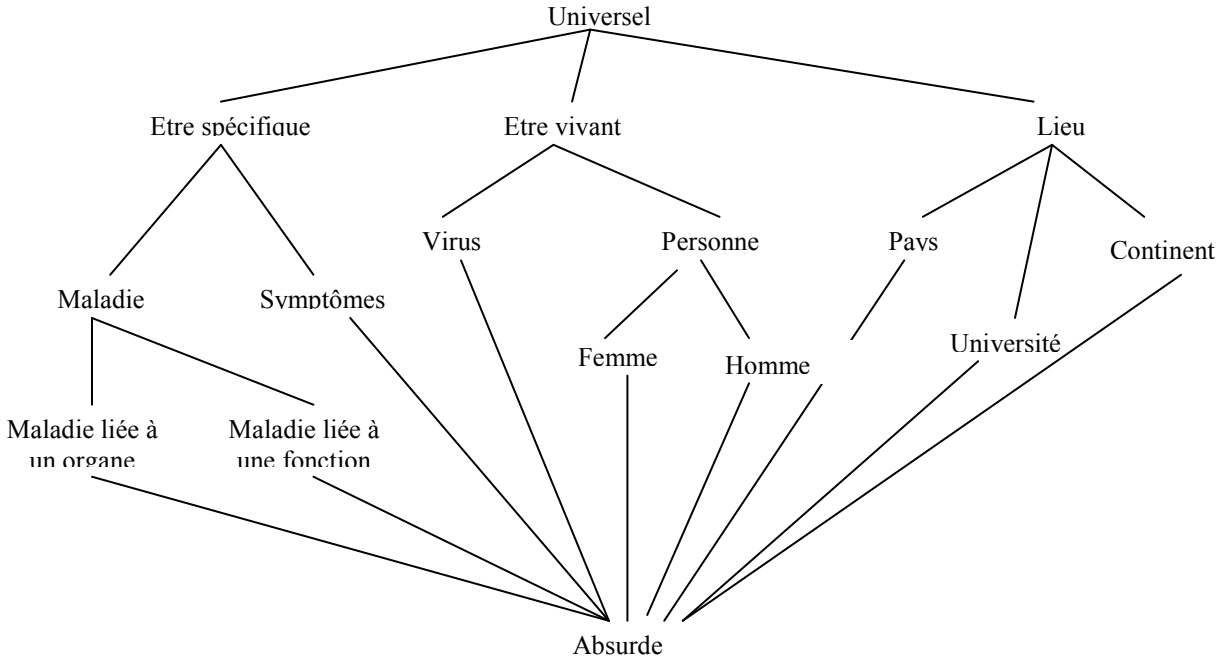

Figure 1. Exemple d'un support d'un graphe conceptuel représenté Sous forme de treillis.

- L'ensemble des types de relations T_r est un ensemble fini et partiellement ordonné par la relation 'sorte de'. Les types de relation entre concepts recensés lors de la conceptualisation des connaissances, permettent d'exprimer la nature des relations qui existent entre des concepts. σ est une application qui associe une 'signature' à tout type de relation. La signature spécifie l'arité du type de relation, ainsi que le type de concept maximal autorisé pour chacun de ses arguments. Exemple de type de relation : 'Agent' soit sa signature σ(agent) = (Action, Individus). Cela signifie que le type de relation agent servira exclusivement à mettre en relation deux concepts 'Action' et 'Individus'.

Exemple : La figure 2, représente la hiérarchie des relations notées par leurs signatures.

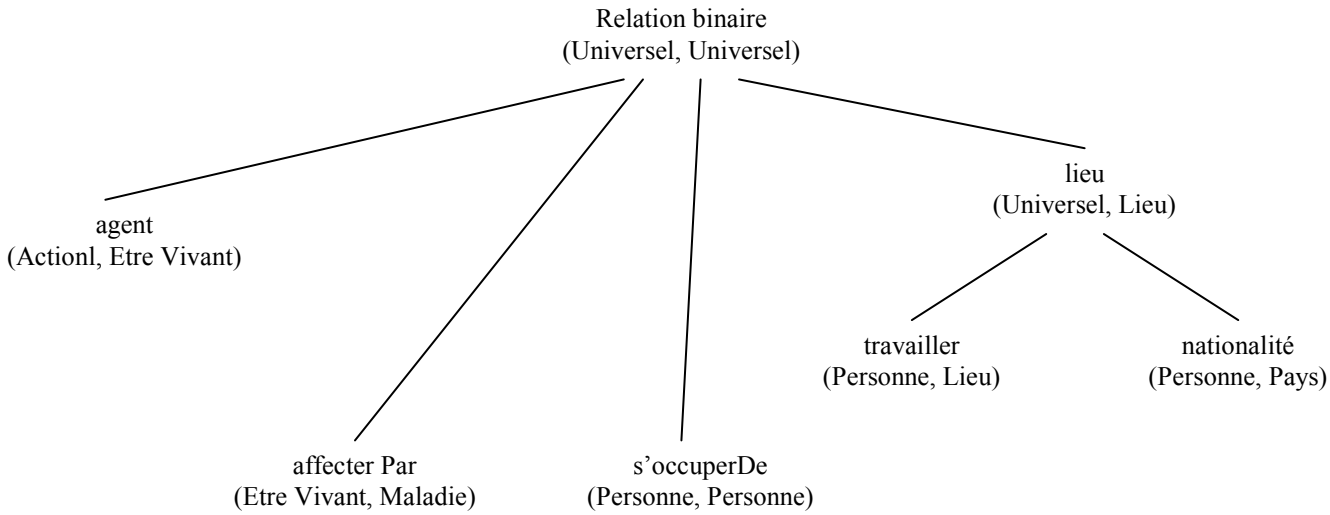

Figure 2. Relations correspondant aux liens entre des types de concepts.

- L'ensemble des marqueurs M, dont la fonction est l'identification des concepts, est composé des marqueurs individuels et d'un marqueur particulier appelé marqueur générique noté *. Le marqueur générique est considéré comme étant plus général que tous les autres marqueurs de M, eux-mêmes étant incomparables deux à deux. Les marqueurs individuels qui appartiennent à l'ensemble M\{*} représentent les instances de concepts. Exemple : « Homme » est une instance de « Humain ».

Le marqueur générique fait quant à lui référence à une instance non spécifiée d'un concept. Il peut être vue comme un article indéfini en langage naturel, ou à une variable quantifiée existentiellement en logique.

II.2. Le niveau assertionnel

La connaissance assertionnelle (les faits) dans une base de connaissances exprimée dans le modèle des graphes conceptuels est constituée par un ensemble de 'graphes conceptuels' qui n'ont de sens que relativement à un support donné.

Un graphe conceptuel G=(R, C, U, Lab) suppose que le support est un multi- graphe, biparti, fini, R et C en sont les deux (02) classes de sommets, respectivement appelées ensemble de sommets relations et ensemble de sommets concepts. U est l'ensemble des arêtes. Les arêtes incidentes à un sommet relation sont totalement ordonnées. Lab est une fonction d'étiquetage pour chaque élément de R, C et U. Chaque sommet relation de R est étiqueté par type(r) qui est un élément de T_r, chaque sommet concept c est étiqueté par une paire (type(c), marqueur(c)) avec $type(c) \in T_c$ et $marqueur(c) \in M \cup \{*\}$, chaque arête u est étiquetée par son rang dans l'ordre total des arêtes adjacentes à un sommet relation donné.

Les sommets concepts, schématisés par des rectangles, représentent les entités, attributs, états, évènements. Tandis que les sommets relations représentent la nature des relations entre concepts.

Exemple

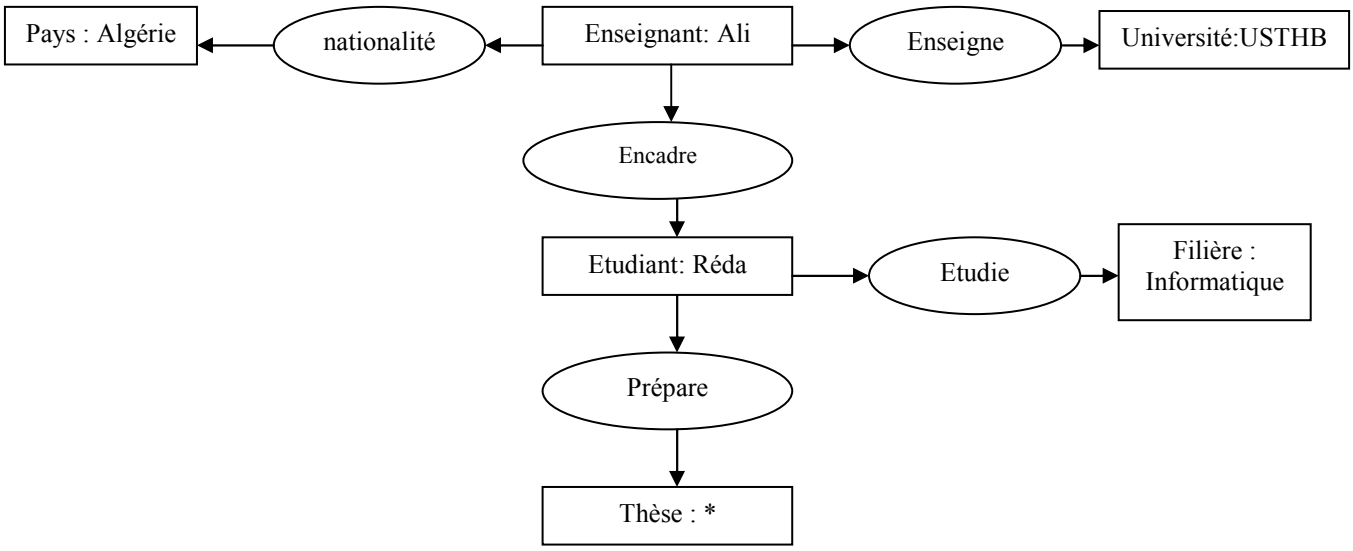

Figure 3. Exemple d'un graphe conceptuel simple

La relation de spécialisation (notée $\leq$) définit un préordre partiel sur l'ensemble des graphes conceptuels. Cette relation de spécialisation peut notamment être calculée par l'opérateur de projection qui est un homomorphisme de graphes autorisant une restriction des étiquettes des sommets. La restriction doit être ici entendu au sens de la relation spécialisation/ généralisation définie dans T_c, T_r et M. G' est une spécialisation de G (noté $G' \leq G$) si et seulement, il existe une projection de G dans G'.

L'opération de projection est une opération fondamentale dans le modèle des graphes conceptuels. Elle permet de vérifier qu'un graphe est une spécialisation d'un autre. Dans une utilisation de type base de données, un graphe conceptuel faisant office de requête permettra de trouver une réponse à chaque fois qu'un graphe plus spécifique sera trouvé dans la base de connaissances assertionnelle.

A noter qu'à chaque graphe conceptuel correspond une formule de la logique du premier ordre. Cette correspondance est obtenue par le biais d'un opérateur Φ défini par Sowa [Sow84].

II.3. Sémantique

Le modèle des Graphes Conceptuels est doté d'une sémantique. Elle est obtenue en traduisant les supports et les graphes en logique du premier ordre (Lpo), grâce à une fonction Φ [Sow84], définie comme suit :

Définition (Sémantique Φ) L'interprétation d'un support S en LPO est donnée par $\Phi(S)$définie comme la conjonction des formules suivantes :

1. $\forall (t,t') \in T_c^2$, $\Phi(t \leq_c t') = \forall x\ t(x) \Rightarrow t'(x)$,
2. $\forall (t,t') \in T_r^2$, $\Phi(t \leq_r t') = \forall x_1, \ldots, x_n\ t(x_1, \ldots, x_n) \Rightarrow t'(x_1, \ldots, x_n)$,
3. $\forall m \in M$, $\Phi(m) = m$.

L'interprétation d'un graphe conceptuel G est donnée par $\Phi(G)$ définie comme la clôture existentielle de la conjonction des formules suivantes :

1. $\forall c \in C$ *avec* $lab(c) = (t, *)$, $\Phi(c) = t(x)$, *où* x *est une nouvelle variable,*
2. $\forall c \in C$ *avec* $lab(c) = (t, m)$ *et* $m \in M$, $\Phi(c) = t(m)$,
3. $\forall r \in R$ *avec* $lab(r) = t$, t *d'arité* n, *et* $c_1, \ldots, c_n$ *les* n *voisins de* t, $\Phi(r) = r(x_1, \ldots, x_n)$, *avec* $\forall i = 1, \ldots, n$, x_i *est une constante ou la variable introduite dans* $\Phi(c_i)$.

Une relation de généralisation entre deux types de concepts $t_1$et t_2 d'un support est traduite en la formule suivante : $\forall x P_1(x) \Rightarrow P_2(x)$ où P_1 et P_2 sont deux prédicats unaires. Une relation de généralisation entre deux types de relation n-aires t1 et t2 d'un support est traduite en la formule suivante :

$\forall x_1 \ldots \forall x_n\ P_1(x_1, \ldots, x_n) \Rightarrow P_2(x_1, \ldots, x_n)$ où P_1 et P_2 sont deux prédicats n-aires.

III. Classification des concepts

Il existe un certain nombre de types de concepts basés essentiellement sur le fait que l'élément représenté (le référent) par le concept est connu ou pas [Sow91].

a. **Concept générique :** Un concept est dit générique si son référent n'est pas connu. Dans ce cas, le concept est considéré comme une interprétation d'un objet existant dans l'univers de discours. Il est non identifiable. Exemple : [Personne], [Université]

b. **Concept individuel :** Un concept est dit individuel, si son référent est connu. Le concept est considéré comme une interprétation d'un objet identifiable. Il est représenté par son référent. Exemple : [Personne : Ali]

c. **Concepts sous forme ensemblistes**

Les noms au pluriel ne nécessitent, en général, qu'une seule lettre supplémentaire dans le langage naturel. Cependant, dans le calcul des prédicats ainsi que dans les graphes conceptuels, ils nécessitent un traitement spécial.

Les référents pluriels dans les graphes conceptuels peuvent être représentés par des ensembles qui permettent des constructions qui n'ont pas lieu dans le langage naturel. La théorie des ensembles, distingue par exemple, l'individu 'Mohamed' de l'ensemble singleton {{Mohamed}} dont le seul élément est {Mohamed}, etc.. qui enfin de compte n'est que l'élément 'Mohamed' à l'intérieur d'un nombre varié d'accolades. Car dans le monde réel, il n'existe pas de marques qui distinguent 'Mohamed' de toutes les autres de ces ensembles qui le contiennent.

Pour cela, plusieurs alternatives [Lus62], [Goo72] et [Bun82] dans la théorie des ensembles ont été proposées. Elles obéissent au principe : « Il n'y a pas de distinction des entités sans distinction de contenu ». Ce même principe doit être utilisé pour définir les référents pluriels dans les graphes conceptuels.

Dans le calcul des prédicats, chaque phrase nominale au pluriel nécessite deux variables quantifiées. La première représente l'ensemble. Tandis que la seconde sert à ranger tous les éléments de l'ensemble. Ces quantificateurs peuvent interagir entre eux de façons complexes.

Exemple

Considérant la phrase : Chaque camion à chariot possède 18 roues.

Le quantificateur universel pour « chaque camion à chariot » a une précédence sur le quantificateur existentiel pour l'ensemble des roues. Voici son graphe conceptuel :

[Camion-Chariot : $\forall$] → (PART) → [Roue :{*}@18]

Par conséquent, le symbole {*} représente des entités non spécifiées de type « Roue », et l'opérateur '@18' précise qu'il en existe 18. L'opérateur Φ translate ce graphe en la formule suivante :

$$(\forall x)[camion\_chariot(x) \Rightarrow (\exists s)(set(s) \wedge count(s,18) \wedge (\forall y \in S)(Roue(y) \wedge part(x, y)).$$

Cette formule a trois quantificateurs : un quantificateur $(\forall x)$ qui range tous les camions à chariot, un quantificateur existentiel $(\exists s)$ pour un ensemble de roues pour chaque x, et un quantificateur universel $(\forall y)$ qui range toutes les roues dans S. Dans ce cas, le quantificateur universel sur Camion-Chariot inclus les deux autres quantificateurs dans sa portée.

Les différents types d'ensemble qu'on peut répertorier sont les suivants :

1) *Ensemble générique :* Le référent est un ensemble, dont les éléments ne sont pas identifiables. En d'autres termes, on ne donne aucune indication sur quels individus il porte. Exemple : [Person :{*}].

2) *Ensemble fini en extension :* On peut énumérer les éléments en les écrivant entre accolades. Les éléments de cet ensemble sont connus. Exemple : [Personne :{Lila, Nawel}]

3) *Ensemble à cardinal spécifié :* On peut préciser le nombre d'élément faisant référence à un concept. Exemple : [Personne :{*}@5] cinq personnes.

4) *Ensemble partiellement spécifié :* Est un ensemble dont on ne connaît pas tous les éléments. Exemple : [Personne :{Lila, Nawel,*}] pour dire qu'il existe des personnes qui sont Lila, Nawel et bien d'autres.

5) *Ensemble collectif Col{*}* : Est un ensemble collectif. Tous les éléments de l'ensemble participent ensemble dans les relations exprimées. Exemple : Comme dans le cas de [Personne : Col{*}@3]←(Agnt)←[Danser] pour dire que les trois personnes dansent la même danse.

6) *Ensemble distributif Dist{*}*: chaque élément de l'ensemble participe séparément dans le modèle des relations exprimées par la phrase comme dans l'exemple suivant :

[DAME : Dist{*}@9]←(Agnt)←[Danse]: il existe une instance du concept « Danser » différente pour chaque « Dame ». Il n'existe pas deux dames qui dansent la même danse.

Un autre exemple d'utilisation simultanée de 'col' et 'Dist', Soit :

[Bloc :Dist{*}@5] ← (PTNT)←[SUPPORT]→ (INST) → [PYRAMID:Col{*}@3]. Pour dire que le support des cinq (05) blocs est constitué chacun de trois (03) pyramides.

7) *Ensemble respectif* : Ce type d'ensemble est utilisé pour exprimer le fait que chaque élément de l'ensemble entretient une relation avec un élément d'un autre ensemble. Exemple : [Homme :Resp{Ali, Réda}] → (FrèreDe)→[Femme :Resp{Nawel, Ratiba}]

IV. Relation conceptuelle

Une relation conceptuelle est une représentation d'une association entre un ou plusieurs concepts. Elle est caractérisée par un libellé qui la classifie et d'un nombre fini d'arcs qui la relie aux concepts associés.

Le rôle principal d'une relation conceptuelle est d'assembler des concepts pour construire une phrase, une idée ou une proposition.

Elle est caractérisée par :

- Une valeur qui est le nombre d'arcs.
- Parmi les n-arcs d'une relation de valence n, (n-1) sont orientés vers la relation (arcs sources) et seulement le $n^{ième}$ est orienté vers un concept (arc de destination).

Il existe des types de relation prédéfinis :

- ***Agent (Agnt)*** qui relie une action [Action] à un concept acteur [ACTEUR]. Ali va à l'école [Ali] ← (Agnt) ← [PARTIR] → (Dest) → [ECOLE]

- ***Expérienceur (EXPR)*** : relie l'état [ETAT] à [ANIME] qui représente le concept qui ressent cet état. Exemple : Ali a froid, sentir le froid est une relation qui relie l'animateur à l'état. [Person : Ali] ← (Expr) ← [Froid]

- ***Instrument (INST)*** relie [ENTITE] à [ACTION]. Donc, relie l'entité à l'action où elle est impliquée de manière causale. Exemple : L'imprimante imprime l'article, l'imprimante est l'instrument [Outil : Imprimante] ← (INST) ← [IMPRIMER]

- ***Objet (OBJ) :*** relie [ACTION] à l'entité sur laquelle porte l'action. Exemple : l'auteur a écrit l'article, Article : Objet. [Auteur : #] ← (Agnt) ← [ECRIRE] → (Obj) → [Article : #].

- ***Manière (MANR)*** : les adverbes représentés par des concepts sont reliés à un autre concept par une relation (MANR) : Les dames dansent gracieusement [Dame :{*}] ←(Agnt) ←[DANSER] → (MANR) → [Gracieux]

– ***Statement (STMT) :*** sert à faire des déclarations :

– ***Progressive (Prog), durée (Dur), Point de temps (PTNT), Passé (PAST), Successeur (Suc) :*** La représentation du temps se fait grâce à ces relations liant des concepts de types [Temps : #Maintenant] et [Temps : #s-time].

Exemple : [Situation : x] →(PTNT) →[Temps] → (Succ)→[Temps :#s-time]. « x » a lieu à un temps passé qui précède le temps de discours [Temps :#s-time].

– ***Stat :*** sert à déclarer des états. Exemple d'une déclaration : Ali déclare qu'il possède une dinde : est une déclaration. Elle est représentée par : [Personne : Ali] →(Stat)→[Posséder]→(PTNT)→[DINDE]. Depuis que « posséder » est un verbe statique, Ali n'est pas l'agent de la possession (ce n'est par une relation de type Agent). Ali est en 'état' de 'Possession', et la 'Dinde est un patient (PTNT) de la possession.

– ***Après (AFTER)*** sert à lier des situations ordonnées.

– ***Donc (ERGO)*** lient des propositions et leurs résultats.

– ***Nécessite (NECS) :*** représentant un lien de nécessité

– ***Obligation (OBLG)*** : représentant un lien d'obligation

Remarques

- Les concepts [ETAT], [ANIME], [ACTION], [ENTITE], …etc. sont des types de concepts prédéfinis.
- Les concepts dont les référents sont écris par « # » représente un référent qui sera défini suivant un contexte donné.

V. Graphes emboîtés et représentation de contexte

L'introduction de graphes emboîtés dans le modèle de base vise à permettre la représentation de connaissances contextuelles. Il s'agit de pouvoir représenter les connaissances à différents niveaux de description, et de pouvoir augmenter ou diminuer la finesse de représentation des connaissances représentées [Sow91].

V.1. Contexte

Un contexte est représenté par un concept avec un ou plusieurs graphes conceptuels emboîtés à l'intérieur du champ référent. Linguistiquement, les contextes reflètent l'emboîtement des clauses dans le langage. Logiquement, ils décrivent des graphes existentiels de Peirce.

Comme dans les graphes de Peirce, l'implication dans les graphes conceptuels est représentée par l'emboîtement de deux contextes négatifs. Un cadre marqué avec le symbole '¬' est utilisé pour représenter un contexte négatif. Un contexte 'If- Then' est représenté de la même manière.

Exemple : Soit à dessiner le graphe correspondant à la phrase suivant : « Si un fermier x possède un âne y alors x bat y.

$$IF[Fermier:*x] \rightarrow (STAT) \rightarrow [Posséder] \rightarrow (PTNT) \rightarrow [ANE:*y]$$
$$THEN[*x] \leftarrow (AGNT) \leftarrow [BATTRE] \rightarrow (PTNT) \rightarrow [*y].$$

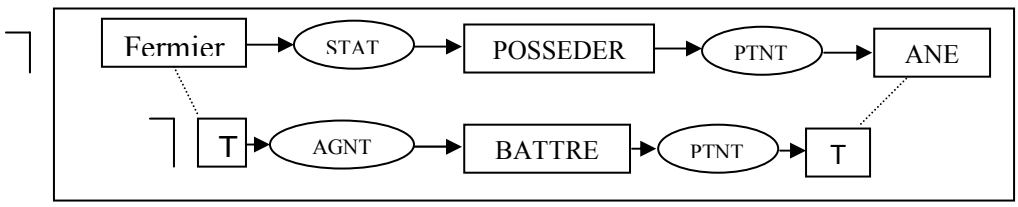

Figure 4. Graphe conceptuel représentant la phrase « si un fermier x possède un âne y alors x bat y

Les graphes conceptuels emboîtés à l'intérieur d'un contexte sont le référent de ce concept. Déjà dans le sens strict, un graphe conceptuel peut seulement avoir lieu comme littéral dans le champ du référent d'un concept de type 'Graphe'.

Pour la définition de contexte, le concept peut être de deux types :

- Concept de type proposition : il représente la proposition déclarée par le graphe. Des phrases contenant des verbes comme 'penser' et 'savoir' définissent des propositions.

- Concept de type situation : il représente une situation décrite par la proposition déclarée par le graphe. Les phrases déclarant des localisations, le temps, des verbes comme 'vouloir', 'craindre', etc.. sont liés aux contextes de type 'Situation'.

Exemple. Soit la phrase : « Ali croit que Lila veut épouser un marin ». Ce qu'Ali croit est une proposition, mais ce que Lila veut est une situation. Il est à noter que 'Croire' et 'Vouloir' déclarent l'existence d' experienceurs (EXPR) au lieu d'une relation « agent ».

Le graphe correspondant est alors :

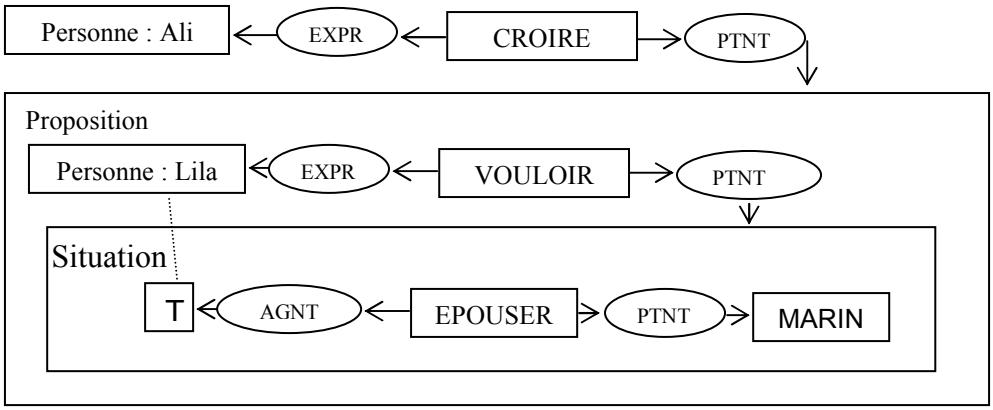

Figure 5. Graphe conceptuel représentant la phrase
« Ali croit que Lila veut épouser un marin »

Depuis que les travaux sur le langage naturel dépendent du contexte, et que le calcul des prédicats ne le prend pas en compte. Beaucoup de phrases ne peuvent être translatées directement au calcul des prédicats. Les graphes conceptuels par contre, adaptent les dépendances contextuelles avec les référents indexicaux faites par le symbole '#'. Le principal indexical est l'article défini 'le', comme dans l'exemple suivant :

« Le chat sur le tapis » : [Chat :#] → (Sur) → [Tapis :#]

Le marqueur # montre que le référent sera connu par rapport au contexte, mais il n'est pas encore résolu par le graphe courant.

D'autres indexicaux peuvent être définis par des expressions comme 'Je' qui réfère à la personne qui parle, et 'tu' ou 'vous' qui réfèrent à la personne à qui on s'adresse.

Le temps est aussi considéré comme 'indexical'. Il réfère au temps quand on parle, le temps de la situation dont on parle, ainsi que les temps composés du passé ou du futur.

Pour simplifier le passage du langage naturel aux graphes conceptuels, des abréviations courtes ont été attribuées à des termes indexicaux communs. Les pronoms 'je' et 'vous ou 'tu' sont remplacés par [Personne :#je], [Personne :#tu] et [Personne :#vous]. 'Ici' et 'maintenant' sont remplacés respectivement par [PLACE :#Ici], [Temps :#Maintenant]. Le temps passé peut être représenté par une relation (PAST), qui s'applique à la situation dont on parle. Cette relation est définie en termes de temps de discours et le contexte courant, représenté par #s-time. La relation (PTIM) représente un point de temps.

Cependant, les graphes conceptuels sont en premier une représentation sémantique. Mais, la syntaxe est importante pour beaucoup d'applications. Les annotations syntaxiques peuvent être additionnées aux noeuds de concepts et de relations séparés de l'information sémantique par un point virgule.

V.2. Annotations syntaxiques

Les graphes conceptuels traduisent une forme logique. Ils représentent seulement le contenu propositionnel de la phrase. Les caractéristiques syntaxiques de la phrase originale sont perdues.

Exemple. Les phrases 'le chat a mangé le poisson' et 'le poisson est mangé par le chat' sont toutes les deux représentées par le même graphe.

(PASSE)→[SITUATION : [CHAT :#]←(Agnt)←[MANGER]→(PTNT) → [POISSON:#]]

Cette forme capte les relations logiques, mais perd l'information concernant l'accentuation. On ne peut pas déterminer le sujet du complément, etc.. .

Pour préserver les nuances syntaxiques. Il est possible d'ajouter des notations syntaxiques qui ne sont pas inclues dans le contenu propositionnel. On peut ajouter '*n*', '*v*' indiquant les parties de discours respectivement le 'nom' et 'le verbe'. '*Subj*' et '*obj*' indiquent les relations '*Agnt*' et '*PTNT*' qui sont exprimées syntaxiquement comme sujet et objet, ...etc. Au moment du traitement des graphes conceptuels, les informations écrites après un point virgule sont ignorées. Elles sont considérées comme *annotations syntaxiques.*

$$(PASSE) \rightarrow [SITUATION : [CHAT : \#; n] \leftarrow (Agnt; Subj) \leftarrow [ETAT; v] \rightarrow (PTNT; Obj) \rightarrow [POISSON : \#; n]]$$

VI. Passage langage naturel au graphe conceptuel

Comme le calcul des prédicats, les graphes conceptuels sont un système formel avec un minimum de caractéristiques incorporées. Le principe de base est que les mots du langage naturel, sont translatés en concepts. Les noms de fonctions comme les prépositions et les conjonctions sont translatés en noeuds relations. Dans ce que suit, on présente quelques cas particuliers :

- Les noms ordinaires, adjectifs, et les verbes sont translatés en étiquettes dans les nœuds concepts : Dame : [Dame], Danser : [Danser] ; heureux : [Heureux].
- Les noms propres sont translatés en champs de référents de concepts dont le type du champ spécifie le type : Minou : [Chat : Minou], maison blanche : [Bâtiment : Maison Blanche]
- Les symboles « # » est utilisé dans le champ de référent pour les références définis par le contexte. Le chat : [Chat :#] ; cela : [T : #Cela].
- Les noms au pluriel sont représentés par le référent au pluriel {*} suivi par un concept optionnel. 9 dames : [Dame :{*}@9], dames : [Dame :{*}]
- Les auxiliaires modaux comme ‘can ou pouvoir’ et ‘must ou devoir’ sont translatés en relations conceptuelles comme « PSBL » (possibilité) ou (OBLG) (obligation) attaché au contexte englobant le graphe de la phrase ou la clause. La phrase ‘Tom peut partir’ devient alors : (PSBL)→[Proposition :[Person : Tom] ← (Agnt) ←[PARTIR]]

  La relation « ¬ PSBL » doit être définie comme une combinaison des relations de négation notée « ¬ » et la relation « PSBL ».
- Les verbes de temps (et les aspects) sont translatés aux relations comme PAST (passé) et PROG (progressive). Ces relations sont définies en termes de relations plus primitives comme DUR (Durée), PTIM (point à temps) et SUCC (successeur) aussi bien que les références définies contextuellement comme #Now (#maintenant), #s-time.

  La phrase : Tom est parti :

  (PAST) → [SITUATION : [PERSON :TOM)←(AGNT)←[GO]]
- ‘être’ quand il est employé comme ‘auxiliaire’ change le temps. Mais quand il est utilisé comme un verbe principal, il montre un lien de coréférence entre les concepts : Mary est enseignante [Person : Mary] ---[Enseignant :#]

La ligne en pointillée reliant deux nœuds concepts montre que ces derniers sont coréférents, c'est-à-dire, ils réfèrent au même individu.

- Le verbe ‘avoir’ quand il est employé comme ‘auxiliaire’ affecte le temps. Mais quand il est utilisé comme ‘verbe’ principal, il peut être translaté en diverses relations comme PART (avoir comme partie) ou POSS (possession).

  La phrase ‘la voiture a un moteur devient’ [Voiture :#] → (PART)→[Moteur].

  Ali a une voiture : [Personne : Ali] → (Poss) → [Voiture :#].

- Les conjonctions comme ‘quand’ ou ‘parce que’ sont translatées en une relation entre les graphes représentant un contexte.

La phrase : Ali est venu après ton départ :

(PASSE) → [[Personne : Ali] ← (Agnt) ← [Venir]] → (APRES) → [Personne :#tu] ← (Agnt) ← [PARTIR]]

- Les relations de temps comme ‘Après (AFTR) et quand (WHEN) lient des situations, mais ERGO (donc) lient des propositions. Pour simplifier les diagrammes, les étiquettes types SITUATION et PROPOSITION peuvent être omises quand elles sont implicites dans les relations.

Il est à noter que ces règles de passage n’étudient en aucun cas l’ambiguïté linguistique entre obligation OBLG et la nécessité NECS. Ces ambiguïtés sont étudiées dans [Sow88]

VII. Opérations élémentaires sur les graphes conceptuels

A la base de toute opération de raisonnement avec des graphes conceptuels se trouve les notions complémentaires de ***spécialisation*** et de ***généralisation*** d’un graphe. Un graphe G est une spécialisation (respectivement généralisation) d’un graphe H si on peut passer de H à G par une séquence d’opérations de spécialisation (respectivement généralisation). H est alors une généralisation de G (on note $G \leq H$). On peut déduire un graphe H d’un graphe G si H est plus général que G. L’ensemble des opérations de spécialisation et de généralisation est le suivant [Hae04] :

VII.1. Opérations de spécialisation

1. **Simplification de relation :** il est possible de supprimer une relation liant les mêmes sommets qu’une autre relation de même type.

2. **Restriction de relation :** il s'agit de remplacer une relation d'un certain type par une autre d'un sous type du premier. En d'autres termes, c'est le remplacement d'un sommet relation de type t_r par un sommet relation de type t'_r tel que $t'_r \leq_r t_r$, en conservant les arêtes ;

3. **Restriction de concept :** un sommet de concept de type t_c et d'identifiant m est remplacé par un sommet concept de type t'_c et d'identifiant m' tels que $t'_c \leq_c t_c$ et $m = *$, ou $m=m'$.

4. **Joint Interne :** deux sommets concepts du même graphe, de même type t_c et de même identifiant m sont réunis dans un même sommet d'identifiant m et de type t_c.

5. **Joint Externe :** deux sommets concepts de deux graphes, de même type t_c et de même identifiant m sont réunis dans un même sommet d'identifiant m et de type t_c.

VII.2. Opérations de généralisation

1. **Duplication de relation :** Opération inverse de l'opération de simplification de relation. Elle duplique le sommet relation ainsi que ses arêtes.

2. **Duplication de concept :** duplication d'un sommet concept et répartition de ses arêtes sur les deux sommets concepts obtenus.

3. **Augmentation de relation :** il s'agit de remplacer une relation d'un certain type par une autre d'un super type. En d'autres termes, c'est le remplacement d'un sommet relation de type t_r par un sommet relation de type t'_r, tel que $t_r \leq_r t'_r$ en conservant les arêtes.

4. **Augmentation de concept :** un sommet de concept de type t_c et d'identifiant m est remplacé par un sommet concept de type t'_c et d'identifiant m' tels que $t_c \leq_c t'_c$ et $m' = *$ ou $m'=m$.

VIII. Modèle des graphes conceptuels flous

Le formalisme logique est une bonne approche pour l'expression humaine et le raisonnement. Le graphe conceptuel et la logique floue sont deux formalismes logiques répondant davantage au langage naturel. Dans la combinaison de ces deux formalismes, le graphe conceptuel fournit une structure des formules qui est près du langage naturel tandis que la logique floue fournit une méthodologie pour traiter certains mots.

Dans ce qui suit, nous commençons par donner quelques notions sur base sur la logique floue.

VIII.1. Logique floue

La logique floue est née en 1965 des travaux de Lotfi A. Zadeh [Zad65], qui a éprouvé le besoin de formaliser la représentation et le traitement de connaissances imprécises ou approximatives, afin de pouvoir traiter des systèmes d'une grande complexité dans lesquels sont, par exemple, présent des facteurs humains. La logique floue intervient dans la manipulation de connaissances imparfaites.

Les connaissances dont on dispose sur un domaine quelconque, pris au sens d'un ensemble d'éléments mis en relation les uns avec les autres, sont généralement imparfaites. D'un côté, parce qu'elles sont obtenues par observations humaines ou instrumentales, généralement soumises à des erreurs, des imprécisions, des incertitudes, et sont par la suite représentées par le langage naturel, la logique, des formules mathématiques, …etc. En d'autres termes, plus le système à étudier est complexe, plus la perte d'information est grande en étape d'observation aussi bien qu'en étape de représentation.

D'un autre côté, la seconde raison d'imperfection est l'absence de rigueur ou due à une flexibilité inhérente au système lui-même et à son fonctionnement. On peut voir cela dans les phénomènes naturels, comme la taille d'une personne, le passage progressif ou non strict du jour à la nuit.

Les imperfections dans les connaissances sont de nature différentes [Bou95]:

1. Les incertitudes concernent un doute sur la validité d'une connaissance. Elle peut être aussi présente lorsqu'on donne intentionnellement des informations erronées ou encore, lors d'une difficulté dans l'obtention ou la vérification de la connaissance. Fiabilité relative par l'intermédiaire d'observateur peu sûr de lui ou susceptible de commettre des erreurs engendre aussi de l'incertitude.

Exemple. « Je crois que la voiture est proche » est une connaissance incertaine.

2. Les imprécisions correspondent à une difficulté dans l'énoncé de la connaissance, soit parce qu'elles sont numériques mal connues, ou exprimées en langage naturel, utilisées pour qualifier une caractéristique de façon vague.

Exemples

- « La taille d'un adulte est entre 1.5 mètres et 2 mètres » est une connaissance flexible.
- « Température douce, grand appartement »est une expression spontanée de connaissance.

- Utilisation de catégories aux limites mal définies comme « Enfant », « Adulte », « Vieillard ».

3. Les incomplétudes sont des absences de connaissances ou connaissances partielles.

Exemple : généralement, il travaille tous les jours.

Ces trois types de connaissances sont dépendantes les unes des autres. Les incomplétudes entraînent des incertitudes, les imprécisions peuvent être associées à des incomplétudes.

Exemple

Soit l'information suivante : « on cherche une personne d'environ 30 ans ». On ne peut pas dire que c'est la même chose que « on cherche une personne de 30 ans » ou encore « on cherche une personne dont l'âge est entre 29 ans et quelques mois et 30 ans et quelques mois ». Le traitement d'une telle information nécessite la définition du terme « environ ».

C'est cet équilibre entre la représentation de l'imperfection et le traitement simple que cherche la logique floue.

La logique floue est l'un des cadres dans lequel sont traitées des imprécisions et des incertitudes, et le cadre dans lequel sont traitées des connaissances numériques et des connaissances exprimées symboliquement par des qualifications du langage naturel.

VIII.1.1. Concept de sous ensemble flou

Dans le but d'éviter le passage brusque d'une classe à une autre, et d'autoriser des éléments à une appartenance partielle à l'une ou l'autre, le concept de sous ensemble flou a été introduit.

La définition de tel sous ensemble vient pour répondre au besoin de représenter des connaissances imprécises exprimées en langage naturel ou obtenues avec des instruments d'observation.

L'idée principale derrière le caractère graduel des sous ensemble flou est que « plus on se rapproche de la caractérisation typique de la classe, plus l'appartenance à cette classe est forte ».

La notion des sous ensembles flous permet le traitement des :

- Catégories aux limites mal définies (comme « ancien », « jeune », …).
- Situations intermédiaires entre le tout et rien (« presque noir », « presque plein »).

- Passages progressifs d'une propriété à une autre (« jeune » à « vieux », de « proche » à « éloigné »).
- Valeurs approximatives (« environ 30 ans »).
- Classe en évitant l'utilisation arbitraire de limites rigides (Difficile de dire qu'une personne de 20 ans est jeune et qu'une personne de 21 ans et moins jeune).

VIII.1.2. Définition d'un sous ensemble flou

Un sous-ensemble flou A de X est défini par une fonction d'appartenance qui associe à chaque élément x de X, le degré $f_A(x)$, compris entre 0 et 1, avec lequel x appartient à A :

$$f_A : X \rightarrow [0, 1]$$

Dans le cas où f_A ne peut prendre que l'une des deux valeurs 0 et 1, le sous- ensemble flou A est un sous ensemble classique de X. Donc, un sous-ensemble classique est un cas particulier des sous-ensembles flous.

VIII.2. Graphes conceptuels Flous

En ce qui concerne l'introduction du flou dans les graphes conceptuels, Des travaux ont été menés pour le faire. Le premier travail connu est constitué par la thèse de Morton [Mor87]. Il a ajouté, dans la première extension, des référents flous (fuzzy refrerents), des opérateurs flous (Fuzzy operations) aux graphes conceptuels flous en appliquant des incertitudes perceptuelles, linguistiques et propositionnelles équivalent à objet, attribut et information. [Wuw83] a étendu la théorie des graphes conceptuels par la fuzziness sur les relations conceptuelles. Il a défini les règles d'inférence utilisées dans les graphes conceptuels flous. Cependant, [Cao96] ont montré des inconvénients dans les travaux de deux extensions citées ci-dessus. Dans [Wuw83] le degré de compatibilité (Compatibility degree) d'un concept ou d'une relation conceptuelle est défini comme un nombre réel dans un intervalle [0, 1] implique des résultats déraisonnables. L'exemple donné pour montrer les inconvénients dans les travaux de Morton et de Wuwongse est le suivant :

Un concept d'entité flou représenté par [T :x|c] où « T » est un type conceptuel, « x » est un référent et « c » la compatibilité de « x » (par rapport à T).

Soit deux concepts A :$[T_a : a|c_a]$ et B :$[T_b : a|c_b]$ tel que : T_a est plus spécialisé que T_b et $c_a \leq c_b$. Selon leur définition de projection floue, alors $[T_a : a|c_a]$ est une projection de $[T_b : a|c_b]$, c'est-à-dire, qu'il existe une implication du concept A au concept B. Mais cette dérivation n'est pas raisonnable. Soit l'exemple suivant : « Lila n'est pas un garçon. Elle est

une fille ». Selon leur définition du concept d'entité floue, on peut représenter cette phrase par :[Garçon : Lila|0] mais ce concept peut entraîner [Garçon : Lila|1] qui exprime « Lila est un garçon » et c'est un conflit.

Pour pallier à ces derniers, Dans [Cao01], les auteurs ont proposé des ensembles de primitifs (primitive sets). Dans ce qui suit on présentera le formalisme des graphes conceptuels issu de cette proposition.

Afin de construire les graphes conceptuels flous, Vilas Wuwongse et Cao Hoang [Cao01] ont proposé un ensemble de primitif qui sont :

- Ensemble de concepts flous S_{TYP}.
- Ensemble de relations floues S_{REL}.
- Ensemble de référents flous S_{REF}.
- Ensemble de mesures floues S_{MEA} qui permet de représenter les attributs comme la couleur, la vitesse et bien d'autres.
- Ensemble de valeurs de vérité floues S_{COM} : Ensemble défini sur un espace des multivaleurs logiques, constitué de 3 classes :
  - Faits positifs : T (very True, fairly True, True)
  - Faits négatifs: F (very False, False, Fairly False)
  - Faits inconnus: U (unknown)

S_{COM} est alors l'union des sous ensembles T, F et U

De cela, un graphe conceptuel flou n'est autre qu'un graphe conceptuel représenté par un uplet : (V_C, V_R, E, P_C) où :

- V_C : ensemble de nœuds de concepts d'entité et d'attributs, tel que :

- Un concept d'entité flou est un triplet sur $S_{TYP} \times S_{REF} \times S_{Com}$ noté [T : X|c] où T : type conceptuel, x : référent et c : comptabilité de x à T. Ce concept possède un dual noté [Univ\T : ¬c], Univ est type de concept générique

- Un concept d'attribut flou est un triplet sur $S_{TYP} \times S_{REF} \times S_{MEA}$ noté [T : x@m] où T : Type conceptuel, x : référent et m est la mesure de x.

- V_R : Ensemble de nœuds de relations conceptuelles floues : chacune définie sur $S_{REL} \times S_{Com} \times N$ noté par [T|c, n], où : T : est un type de relation conceptuelle. c : est la compatibilité de la relation au type T et n : le nombre de référent participant à la relation.

- E : Ensemble des arêtes orientées du graphe.

- P_C : Partition de tous les noeuds de concept par une relation de coréférence.

Exemple

Soit la phrase suivante : Mohamed est un jeune homme, et c'est vrai qu'il travaille dans une entreprise et elle s'appelle X

$$[Homme : Mohamed] \rightarrow (Travail \mid vrai) \rightarrow [ENTREPRISE \mid Vrai]$$

$$[Homme : Mohamed] \rightarrow (Attribut) \rightarrow [Age : @\, JEUNE]$$

Dans [Cao01] une proposition a été retenue concernant un type flou définit comme composé du type de base et d'une valeur floue.

Exemple

- $[(Etudiant\_USTHB, vrai) : Sara]$ exprime que c'est vrai que Sara et une étudiante de l'USTHB.

- Pour dire que Sara est une étudiante de l'USTHB qui aime porter une montre de couleur Bleu, on écrit :

[(ETUDIANT_USTHB, very true) : Sara]→ (Aime|vrai) → [Montre : *]
↓
[Couleur: @ Bleu]

IX. Conclusion

Une étude détaillée des graphes conceptuels a été faite. La représentation d'expressions de la longue naturelle qui sont parfois floues, de nature vague, imprécise et incertaines, …etc. est faite grâce à l'extension du modèle des graphes conceptuels (GC) par les principes de la logique floue (Fuzzification des graphes conceptuels). Ce qui donnera les graphes conceptuels flous (GCF).

Comme ce modèle de représentation de données est proche d'un des modèles du Web Sémantique RDF(S), l'idée de translater les GCF en RDF(S) et cive versa à travers un ensemble de règles faisant l'objet du prochain chapitre

Chapitre 4

Chapitre IV

Passage Graphes conceptuels flous vers RDF(S)

et vice versa

I. Introduction

Le modèle des graphes conceptuels est un formalisme de représentation proche du modèle de données d'un des langages du web sémantique, à savoir RDF(S). Mais, le langage naturel est connu pour être vague et imprécis. Alors, une extension du modèle des graphes conceptuels pour la prise en compte des informations floues a été proposée, appelé « Les graphes conceptuels floues ».

Dans ce chapitre, on présentera en détail notre proposition de règles de passages des graphes conceptuels flous vers RDF(S) et vice versa après avoir présenter l'extension de la classification des concepts ajoutant une dimension supplémentaire pour prendre en compte le caractère flou des concepts

Pour le faire, on commencera par répertorier les correspondances entre les deux modèles afin de bien cerner les points communs et les différences entre ces derniers.

II. Correspondance entre RDF(S) et les graphes conceptuels

Après une étude détaillée de RDF(S) et les graphes conceptuels nous avons répertorié les points communs suivants :

- Les connaissances terminologiques et assertionnelles sont séparées, dans les deux modèles. La hiérarchie de concepts et de relations des graphes conceptuels est équivalente à celle de classes et de propriétés de RDF et réciproquement.
- Dans les deux modèles, les connaissances assertionnelles sont positives, conjonctives et existentielles.
- Dans les deux modèles, nous retrouvons la représentation graphique qui facilite la visualisation de la connaissance à modéliser :
    - Un arc étiqueté par une propriété « p » dans un graphe RDF, correspond à un sommet relation de type « p » dans un graphe conceptuel.

- Un sommet « ressource » identifié par une URI est traduit par un sommet de concept individuel des graphes conceptuels, dont le marqueur est l'identificateur de la ressource, et dont le type de concept est une classe à laquelle est rattachée la ressource par « rdf :type » dans le graphe RDF.

- Les ressources anonymes de RDF(S) sont représentées par des concepts génériques dans les graphes conceptuels. Le type est connu grâce à sa liaison avec une classe de type « rdf :type ».

Il existe cependant quelques différences que nous mentionnons dans ce qui suit :

- L'arité associée aux relations, est de deux (02) dans RDF et de type n-aire dans les graphes conceptuels. Les ressources intermédiaires (ressources anonymes) peuvent être utilisées pour relier les ressources par des propriétés binaires.

- Les propriétés RDFS peuvent avoir plusieurs domaines grâce à « rdfs : domaine », tandis que les relations des graphes conceptuels ont un domaine unique. Cette multiplicité de domaines d'une propriété RDF peut être traduite dans le modèle des graphes conceptuels par une relation dont le domaine est le concept le moins général des types de concepts traduisant les domaines de la propriété.

Ces différences ne sont en rien un obstacle pour assurer le passage, entre les graphes conceptuels, RDF et RDFS. En effet, ce sont les points communs qui ont été exploités pour définir les règles de passage. On commencera par le passage le plus évident qui est le passage d'un graphe représentant une déclaration RDF(S) vers un graphe conceptuel.

III. Passage RDF(S)/ Graphes conceptuels

Après avoir cité les similitudes et les différences entre les deux modèles, il paraît naturel de s'intéresser :

a. Aux déclarations RDF par leurs traduction par des graphes conceptuels.

b. A la hiérarchie de classes apparaissant dans un schéma RDF, et qui correspond à une hiérarchie de type de concepts dans un graphe conceptuel.

c. A la hiérarchie de propriétés apparaissant dans un schéma RDF et qui correspond à une hiérarchie de type de relations dans un graphe conceptuel.

III.1. Déclaration RDF

Les déclarations RDF de base sont de la forme <Sujet, Prédicat, Objet>. Elles peuvent être écrites sous la forme : Prédicat (Sujet, Objet). Chaque ressource décrite peut être déclarée dans le modèle des graphes conceptuels comme un concept de type 'ressource' dont le référent est l'URI associée à cette ressource dans la déclaration RDF.

Les propriétés des ressources (prédicats des déclarations RDF) sont traduites par des relations conceptuelles.

On considère deux exemples. Dans le premier exemple, on a une seule déclaration pour une ressource. Par contre, dans le second exemple, pour la même ressource, il existe plusieurs déclarations

Exemple 1

En *RDF/ XML* :

```
<rdf :Description about =''http://www.usthb.dz/542'>
    <Nom> Ali </Nom>
</rdf : Description>
```

Cette déclaration s'écrit sous la forme de triplet comme suit <http://www.usthb.dz/542, Nom, 'Ali'> ou encore sous forme de prédicats : Nom ('http://www.usthb.dz/542', 'Ali')

Le graphe conceptuel

[Resource : http://www.usthb.dz/542'] → (Nom) → [Littéral : Ali]

Exemple 2

En *RDF/ XML* :

```
<rdf :Description about =''http://www.usthb.dz/542'>
    <Nom> Ali </Nom>
    <Adress> El Alia </Adress>
</rdf : Description>
```

Le graphe conceptuel

[Resource : http://www.usthb.dz/542'] → {→ (Nom) → [Littéral : Ali]
→ (Adress) → [Littéral : El Alia] }

On peut donc considérer une description RDF comme une instance de type de concepts « Resource » dans les graphes et les propriétés associées comme les relations liées à ces concepts. Le référent de ce concept de type « ressource » est l'URI correspondante à cette ressource.

III. 2. Descriptions RDF emboîtées

Dans des cas particuliers, les déclarations concernant une ressource peuvent avoir des propriétés (prédicats) dont les valeurs ne sont pas atomiques. Le graphe conceptuel correspondant à la déclaration de cette ressource contient à l'intérieur une autre déclaration de concept de type 'Ressource'. En d'autres termes, on aura un graphe emboîté.

Exemple 3

En *RDF/ XML* :

```
<rdf :Description about =''http://www.usthb.dz/542'>
    <Nom> Ali </Nom>
    <Address rdf :Description about ='http://www.usthb.dz/ali'>
        <Num> 32 </Num>
      <Rue> El Alia </Rue>
      <Ville> Alger </Ville>
    </Adresse>
</rdf : Description>
```

Graphe conceptuel :

[Resource :'http://www.usthb.dz/542]-

{ → (Nom) → [Littéral : Ali]

→ (Adresse) → [[Resource :'http://www.usthb.dz/Ali] → (Num) → [Littéral :32]

→ (Rue) → [Littéral :El Alia]

→ (Ville) → [Littéral :Alger]]}

Dans le cas de description emboîtée, la translation consiste alors en la création d'un concept typé 'Ressource' pour chaque description de ressource emboîtée.

La ressource emboîtée http://www.usthb.dz/Ali est liée par le biais de la relation 'Adresse' à la ressource incorporée 'http://www.usthb.dz/542'.

III. 3. RDF Schéma

Les descriptions RDF peuvent être typées suivant l'ontologie prédéfinie appelée RDF Schéma par des déclarations précédées par des noms de domaines. Dans ce cas, un concept d'un type associé au domaine correspondant est crée dans les Graphes Conceptuels.

Exemple 4

En *RDF/ XML* :

```
<rdf : RDF xmlns:rdf=''http://www.w3.org/1999/02/22-rdf-syntax-ns#>
   Xmlns:inst='http://www.usthb.dz/department#'>
<inst: Local  rdf:about 'http://www.usthb.dz/department/local1234>
            <inst :type> Laboratoire  </inst :type>
            <inst :Num> Dix </inst :Num>
        </inst :Local>
</rdf : RDF>
```

Graphe conceptuel :

[Local :http://www.usthb.dz/département/local]-

{→ (type) →[littéral :Laboratoire]

→(Num) → [Littéral : Dix]}

Il est à noter qu'afin de prévoir les conflits de noms pour les différents schémas, on suppose que les concepts ainsi que les relations doivent être préfixés de la même façon, par le même domaine.

III.3.1. Classe :

Les classes RDFS peuvent être modélisées comme des types de concepts dans les graphes conceptuels.

a. Concept de type Resource :

Le sommet de la hiérarchie de type de concept est la « Ressource » du fait que tout dans RDF est une « Resource ». Donc, toute déclaration de type <rdfs :Class ref : ID ='C1'/> est translatée dans les graphes conceptuels en : type Concept C1 $<_c$ Resource pour dire que « C1 » est un sous type du type « Ressource ».

b. Les Sous-classes : Il existe une correspondance claire et directe entre 'rdf :subClassOf' et 'sous_type' dans les GCs.

Exemple 5

En *RDF/ XML* :

```
<rdfs :Class rdf :ID =’Enseignant’>
    <rdfs :subClassOf rdf :resource =’Personnel_Université’/>
</rdfs :Class>
```

La classe des enseignants est une sous classe de la classe ‘Personnel_ Université’. Puisque la classe est modélisée par un type de concept, la représentation d’un sous_type correspond à la définition d’une classe dont le type est un sous_ type de la classe mère.

Graphe conceptuel

Type concept Enseignant $<_c$ Personnel _ université.

III.3.2. Les propriétés et sous propriétés

a. Les propriétés :

On a vu dans le second chapitre qu’une propriété s’applique à une ou plusieurs classes par ‘Domain’ et prend ses valeurs dans une seule classe spécifiée par ‘Range’.

Les propriétés sont translatées en des relations dans les graphes conceptuels. La signature de la relation est comme suit : type-relation ‘Nom’ (Domaine, range). En d’autres termes, le domaine représente la source de la relation et le ‘range’ spécifie la destination de la relation.

Exemple 6

En RDF/ XML

```
<rdf :Property Id=’Enseigne’>
    <rdfs :domaine rdf :resource =’’#Enseignant ‘’/>
    <rdfs :range rdf :resource =’’#Etudiant ‘’/>
</rdf : Property>
```

Graphe Conceptuel

La signature de la relation ‘Enseigner’ est la suivante : Type-relation Enseigner (Enseignant, Etudiant).

Dans RDF, une propriété peut avoir plusieurs classes comme domaine car la signature d’une relation conceptuelle est unique. Le passage d’une propriété RDF avec plusieurs domaines vers les relations conceptuelles ne peut se faire directement. Des solutions existent. Il s’agit de :

- Remplacer le domaine de la relation par une relation ancêtre, si les relations de différents domaines en possèdent une.
- Pour toutes les classes du domaine de la propriété, remplacer par une classe abstraite définie pour ce but,
- Remplacer par le sommet de la hiérarchie des concepts qui relie toutes les ressources. La propriété doit être permise pour toutes les classes.

Exemple 7

En RDF/ XML

```
<rdf : Property ID='Enseigner'>
    <rdfs : domaine rdf :resource ="#Enseignant_ Permanent ''/>
    <rdfs :domaine rdf :resource ="#Enseignant_ vacataire ''/>
    <rdfs :range rdf :resource ="#Etudiant ''/>
</rdf :Property>
```

L'une des solutions consiste à remplacer les deux domaines par une super classe des deux classes représentant le domaine.

Graphe conceptuel

Si Concept type Enseignant :

Concept type Enseignant _ Permanent $<_c$ Enseignant.
Concept type Enseignant _ Vacataire $<_c$ Enseignant.
Type relation Enseigner (Enseignant, Etudiant).

b. Sous propriété

Afin de raffiner les propriétés existantes, comme dans la définition des sous classes, les 'sous propriétés' sont employées.

Si « ρ_1 », « ρ_2 » sont deux propriétés et si $\rho_1 < \rho_2$ et ρ_1(URI, C) alors ρ_2 (URI, C). En d'autres termes, si une URI possède une valeur « C » par « ρ1 » et si « ρ_1 » est une sous propriété de « ρ_2 », alors URI possède la même valeur par la propriété « ρ_2 ».

Exemple 8

En RDF/ XML

```
<rdf : Property ID='Enseigner'>
<rdfs : domaine rdf :resource ="#Enseignant ''/>
    <rdfs :range rdf :resource ="#Etudiant ''/>
```

```
</rdf :Property>
<rdf : Property ID='Donne-TD'>
<rdfs : subProperty rdf :resource ="#Enseigner ''/>
</rdf :Property>
```

En termes de graphes conceptuels, une sous propriété RDF est translatée en un type de relation qui est un sous type de relation translatant la sous propriété.

Graphe conceptuel

Type relation Enseigner (Enseignant, Etudiant)

Type relation Donne -TD $<_r$ Enseigner.

III.4. Translation des propriétés supplémentaires de RDF(S)

- La réification

La réification est une déclaration à propos d'une déclaration. RDF offre un vocabulaire incorporé pour décrire des déclarations RDF. Ce vocabulaire consiste à un type 'rdf :Statement' et les propriétés 'rdf :Predicate' qui identifie la propriété originale, 'rdf :Subject' identifie la ressource décrite dans la déclaration modélisée, 'rdf :Object' identifie la valeur de la propriété et 'rdf :type' représente le type de la nouvelle ressource.

Exemple 9

En RDF/ XML

```
<rdf : Description>
    <rdf :Subject rdf :resource :'http://www.usthb.dz/département/local1234'/>
    <rdf :Predicate : rdf :resource ='http://www.usthb.dz/département/ type'' />
    <rdf : Object rdf :resource = ''http://www.usthb.dz/département/laboratoire''/>
    <rdf :type rdf :resource =''http://www.w3.org/1999/02/22-rdf-syntaxe-ns#Statement''/>
</rdf :Description>
```

Le modèle des graphes conceptuels est muni de moyen d'expression des déclarations imbriquées à savoir les graphes emboîtés via le type de concept 'Statement' et le référent est un type de graphe représentant la déclaration originale.

Graphe conceptuel

[Statement :[Local :http://www.usthb.dz/département/local1234]→(type)→[Littéral:Laboratoire]]

III.4.1. Les Containers

La représentation des ensembles est faite dans RDFS grâce aux containers. Ils offrent un moyen de définition de groupe quand la valeur du triplet est un ensemble.

Il existe trois types de containers: « bag », « sequence » et « Alternative ». ils sont tous munis de la propriété <rdf :li> qui permet d'énumérer les composants de l'ensemble dans chacun des cas.

Exemple 10

En RDF/ XML

```
<inst: Local rdf :about =”http://www.usthb.dz/département/locaux”/>
        <inst: type>
                <rdf:Bag>
                        <rdf :li> laboratoire </rdf :li>
                        <rdf :li> Salle-TD </rdf :li>
                        <rdf :li> Bureau </rdf :li>
                </rdf :Bag>
    </inst : type>
    </inst :Local>
```

Les types des locaux sont: 'laboratoires', 'salle – TD' et 'bureau'. Chacun d'entre eux est précédé par <rdf :li> qui peut être vu comme une propriété qui est translatée à son tour en un type de relation dans les GCs dont la signature est « rdf :li (Container, Resource) ».

L'exemple 10, peut être translaté dans les graphes conceptuels comme suit :

Graphe conceptuel

[Local : 'http://www.usthb.dz/département/locaux] → (type) → [Bag] –
{→ (rdf:li) →[Laboratoire]
→ (rdf:li) → [Salle-TD]
→ (rdf:li) → [Bureau]
}

Il est à noter qu'on a représenté [Bag] sous forme d'un type de concept qui est un sous type du type de concept [Container].

De cette façon, le type de concept abstrait (qui n'a pas d'instances directes) [Container] possède trois sous types qui sont : [Bag], [Séquence] et [Alternative].

III.4.2. 'Container' et 'aboutEach'

Quand une déclaration concerne un groupe d'éléments, l'utilisation de 'aboutEach' sera très utile.

Dans l'exemple précédent si on veut dire que les trois types de locaux sont disponibles à l'institut d'informatique alors on écrit :

Exemple 11

En RDF/ XML

```
<inst: Local rdf :about =''http://www.usthb.dz/département/locaux''/>
    <inst: type>
        <rdf:Bag ID:'TYPE'>
            <rdf :li> laboratoire </rdf :li>
            <rdf :li> Salle-TD </rdf :li>
            <rdf :li> Bureau </rdf :li>
        </rdf :Bag>
</inst : type>
</inst :Local>

<rdf: Description aboutEach:#TYPE''>
    <inst:Dispo> INFORMATIQUE </inst:Dispo>
</rdf :Description>
```

Graphe conceptuel

[Local : 'http://www.usthb.dz/département/locaux] → (type) → [Bag] –
{→ (rdf:li) →[Laboratoire] →(Dispo)→[Informatique]
→ (rdf:li) → [Salle-TD] →(Dispo)→[Informatique]
→ (rdf:li) → [Bureau] →(Dispo)→[Informatique] }

Le passage de RDF(S) vers les graphes conceptuels est un passage plus ou moins évident du fait qu'il est basé sur les ressemblances entre les deux modèles. Cette évidence a lieu parce qu'on se base plus sur une compréhension des déclarations RDF(S) plutôt que sur une compréhension des déclarations exprimées en langage naturel. Le passage inverse est bien sur moins évident.

IV. Passage des graphes conceptuels flous vers RDF(S)

Afin de réaliser le passage des graphes conceptuels flous vers RDF(S) cas par cas, les règles de passage doivent prendre en compte la classification des concepts proposée par [Sow91] déjà présenté dans le chapitre 3 et que cette même présentation peut être étendue pour la prise en charge du caractère flou des concepts donnés dans [Cao01].

On changera légèrement cette classification selon que le référent soit connu ou pas, qu'il corresponde à un seul élément ou à un ensemble.

IV.1. Classification des concepts avec introduction du flou

IV.1.1. Concept générique

Un concept est dit générique si son référent n'est pas connu. Dans ce cas, le concept est une interprétation d'un objet existant de l'univers de discours mais il est non identifiable.

Exemple : [Personne], [université]

Du fait que le référent soit inconnu, la compatibilité du référent au type de concept dépend essentiellement de celui-ci. On peut dire, donc, que la compatibilité est inconnu ($U \subset S_{COM}$). On peut alors l'écrire : [Personne : * |Unknown], [Université : * |Unknown].

IV.1.2. Concept individuel

Un concept est dit individuel, si le référent est connu. Le concept est une interprétation d'un objet identifiable et représenté par le référent.

Exemple : [GrandHomme : Réda| vrai] où Réda désigne un grand homme.

Dans ce cas, le référent est connu. La compatibilité du concept à son type conceptuel est aussi connue. Ce concept est alors soit un concept d'entité flou, soit un concept d'attribut flou suivant que celui-ci peut être mesuré ou non.

IV.1.3. Concept ensembliste

Dans les cas où le référent n'est pas un seul élément mais un ensemble, la compatibilité de l'ensemble représentant le référent au type conceptuel dépend aussi du concept [Sow91]. Le degré de compatibilité ou la mesure de compatibilité (suivant le cas) sont les mêmes pour tous les éléments de l'ensemble. Les cas possibles de ces référents sont les suivants :

- *Ensemble générique.* Le référent peut être un ensemble dont les éléments ne sont pas identifiables.

  Exemple : [Personne :{*}] désigne plusieurs personnes. Dans un graphe conceptuel flou, cette écriture serai alors [Personne : {*}|C] où C est la compatibilité de l'ensemble au type conceptuel, qui peut être « Unknown » aussi.

- *Ensemble de cardinal spécifié.* On peut préciser le nombre d'éléments faisant référence à un concept.

Exemple : [Personne :{*}@5 |C] pour dire cinq Personnes ayant une compatibilité C à ce concept.

- *Un ensemble fini en extension.* On peut énumérer les éléments en les écrivant dans un ensemble (entre accolade).

  Exemple : [PersonnePresent : {Ali, Reda, Zoheir}|C] pour dire que les personnes présentes sont : Ali, Réda et Zoheir et sont compatibles avec un degré C.

- *Un ensemble partiellement spécifié.* C'est un ensemble dont on ne connaît pas tous les éléments.

  Exemple *:* [Personne :{Ali,*}|C] pour dire « Ali et bien d'autres ».

- *Les ensembles collectifs.* Ces ensembles sont utilisés pour dire que le reste du graphe conceptuel est appliqué à l'ensemble des éléments.

  Exemple *:* Pour dire qu'il y a trois personnes qui possèdent des enfants, on écrit [Personne : Col{*}@3|C] → (Possède) → [Enfant]

- *Les ensembles respectifs.* Ils sont utilisés pour exprimer le fait que chaque élément de l'ensemble entretient une relation avec un élément d'un autre ensemble.

  Exemple *:* [Personne : Resp{Ali, Réda}|C1] → (Frèrede)→[Fille:Resp{Nawel, Ratiba}|C2]. [Personne : Resp{Ali, Réda }| C1] sont respectivement les frères de Nawel et Ratiba.

Remarque :

Dans les cas exemples, ci-dessus, nous avons considéré des concepts précis où C = Vrai.

Le formalisme ainsi modifié permet de représenter les concepts et les relations vagues et imprécises dans les différents cas. Notre but est de ramener une telle représentation au web sémantique. Pour cela, il est nécessaire de rapporter les points communs entre les Graphes Conceptuels flous et RDF(S) et de donner par la suite l'ensemble des propositions pour le passage des graphes conceptuels vers RDF(S).

IV.2 Passage graphes conceptuels Flous→ RDF(S)

Il paraît clair que les deux types de représentation (RDF, graphes conceptuels classiques et flous) présentent des points communs surtout en ce qui concerne les concepts.

Un concept dans les graphes conceptuels, classiques et flous, correspond à une ressource RDF ou à une classe RDF(S) ayant un type. La hiérarchie des concepts est définie grâce à «RDFS :subClassOf». D'un autre côté, une relation dans les graphes conceptuels devient une propriété RDF dont la précision de la source et la destination est faite grâce à «rdfs :Domain» et «rdfs :range».

Dans les graphes conceptuels flous, les concepts sont soit des concepts d'entité floue, soit des concepts d'attribut flou. La représentation des deux doit donc être différente.

IV.2.1 La représentation des concepts d'entité floue dans RDF(S)

La reformulation de la signification de cette notation de concept [T :x|c] est la suivante : Le concept, dont *le référent* est x, a une compatibilité c avec *un type de concept T.*

Le concept est une ressource RDF dont le type est la Super-classe associée et le référent est une instance possible. La compatibilité peut être représentée par une propriété associant une valeur à une instance. A ce niveau, il faut définir une classe pour le type de concept. Le référent est alors une instance de ce concept.

Exemple

Soit le concept [Homme : Réda|Vrai]. Dans ce cas, on va simplement créer une classe Homme. Réda ne sera alors qu'une instance de la classe Homme.

La compatibilité est de 3 types. La super-classe Compatibilité a trois (03) sous-classes qui sont classe positive, classe négative et classe inconnue. En RDF/ XML on écrira :

```
<rdfs : Class rdf :ID=''ConceptEntitFlou''>
<rdfs : Class rdf :ID=''Compatibility''>
<rdfs:Class rdf:ID=''Positive">
      <rdfs:subClassOf  rdf:resource=''#Compatibility''/>
</rdfs :Class>
<rdfs:Class rdf:ID='Negative'>
       <rdfs:subClassOf  rdf:resource=''#Compatibility''/>
</rdfs :Class>
 <rdfs:Class rdf:ID='Unknown'>
      <rdfs:subClassOf  rdf:resource=''#Compatibility''/>
```

```
</rdfs :Class>
```

Le degré de compatibilité du réfèrent au concept est connu grâce à une relation « Estcompatible » comme suit :

```
<rdfs:Property rdf:ID=''Estcompatible''>
        <rdfs:Domaine rdf:resource=''#ConceptEntitFlou''>
        <rdfs:Range rdf:resource =''#Compatibility''>
</rdfs:Property>
```

Sachant que la classe «compatibilité» possède trois sous classes à savoir «positive», «négative» et «Unknown», on définit alors pour chaque cas une propriété.

```
<rdfs:Property rdf:ID=''PositiveCompatibilité''>
      <rdfs:subPropertyOf   rdf:resource ='#Estcompatible'/>
     <rdfs:Domaine rdf:resource=''#ConceptEntitFlou''>
      <rdfs:Range rdf:resource =''#Positive''>
</rdfs:Property>
<rdfs:Property rdf:ID=''NegativeCompatibilité''>
       <rdfs:subPropertyOf  rdf:resource ='#Estcompatible'/>
      <rdfs:Domaine rdf:resource=''#ConceptEntitFlou''>
      <rdfs:Range rdf:resource =''#Negative''>
</rdfs:Property>
<rdfs:Property rdf:ID=''InconnuCompatibilité''>
     <rdfs: subPropertyOf  rdf:resource ='#Estcompatible'/>
     <rdfs:Domaine rdf:resource=''#ConceptEntitFlou''>
     <rdfs:Range rdf:resource =''#Unknown''>
</rdfs:Property>
```

IV.2.2. La représentation des concepts d'attribut flou dans RDF(S)

Pour les concepts d'attributs flous, on peut appliquer la même idée. Seulement, les mesures de compatibilité ne sont plus des valeurs qui appartiennent à la classe compatibilité, mais une classe qui contient des mesures subjectives.

A ce niveau, on a vu que la représentation de la compatibilité est facile par RDF(S). Il est important, maintenant, de spécifier comment représenter les types de concepts et les relations déjà définis dans la section précédente.

IV.2.3. Les Concepts

IV.2.3.1. Concept générique

Un concept est générique, si son référent n'est pas connu. En RDF/XML, on peut définir une classe pour les concepts génériques et instancier par la suite les objets de cette classe par référent défini.

Exemple : Soit le concept générique [Personne]. En RDF, on peut l'écrire comme suit :

< rdf :Description rdf :about =''http://www.université.dz/Personne''>

<rdf:Type rdf :resource=''http://www.w3.org/TR/rdf-Schema#Class''>

</rdf:Description>

Ou encore, on l'écrit :

<rdfs :Class rdf :about=''://www.université.dz/Personne''/>

IV.2.3.2. Concept individuel

Dans ce cas le référent est connu. Le concept correspond à l'interprétation d'un objet identifiable. Il s'agit en RDF de la déclaration d'une classe représentant le type de concept associé au référent (l'élément identifiable) et il restera à déclarer par la suite un élément de cette classe correspondant au référent lui-même.

Exemple : On peut déclarer John Smith comme une personne et cette personne est le créateur de la page web particulière. Le concept individuel dans cette déclaration est [Personne : John Smith]. En RDF, on peut l'écrire comme suit :

```
< rdf : RDF
    xmlns : rdf= ''http:// www.w3.org/1999/02/22-rdf-syntax-ns#''>
    xmlns : dc =''http://purl.org/metadata/dublincore-core#''>
<rdf :Description about :'http:// www.ourorg.net/metadata.html''>
<dc:Creator>
  <rdf :Description rdf:about =''http://www.ourorg.net/member/JohnSmith>
    <rdf:Type rdf :resource =''http://www.université.dz/Personne''/>
```

```
   </rdf:Description>
</dc:Creator>
</rdf: Description>
</rdf:RDF>
```

IV.2.3.3. Concepts ensemblistes

- *Ensemble générique.* Le référent dans ce type de concept est un ensemble non identifiable. Pour ce type de déclaration, on peut utiliser les possibilités de RDF dans le cas de déclaration d'ensembles. Et puisque dans les ensembles génériques, il n'y a aucun ordre à respecter, l'utilisation des containers (type Bag) est recommandée car le nombre d'éléments de l'ensemble n'est pas limité au préalable.
- *Ensemble de cardinal spécifié*. Le référent de ce type de concept peut être un ensemble dont on connaît déjà le nombre d'éléments le constituant. On peut dans ce cas, utiliser comme dans le cas précédent les possibilités de RDF dans la définition de groupes ou d'ensembles en précisant le nombre de nœuds pouvant exister. Cet ensemble sera comme une liste chaînée dont la fin est marquée par NIL. Les Collecteurs ont exactement cette fonction de délimitation des ensembles. Et puisque les éléments de cet ensemble ne sont pas identifiables, chaque nœud aura alors comme valeur une URI se terminant par « _n » où n est un entier différent de zéro.

Exemple : Trois personnes est un ensemble dont le cardinal est égal à 3, [Personne : {*}@3]:

```
< rdf : RDF>
        xmlns: rdf=''http://www.w3.org/1999/02/22-rdf-syntaxe-ns#''>
        xmlns : gc=''http://www.université/termes#''>
<rdf :Description  rdf :about :'http://www.université/Personne''>
<gc :HasReferent>
  <rdf : ParseType=''Collection''>
    <rdf :Description  rdf:about=''http://www.université/Personne/_ 1''/>
    <rdf :Description  rdf:about=''http://www.université/Personne/_ 2''/>
    <rdf :Description  rdf:about=''http://www.université/Personne/_ 3''/>
</rdf : Description>
</rdf : RDF>
```

- *Ensemble fini en extension.* Dans ce type de concepts, les éléments constituant l'ensemble peuvent être énumérés.

Exemple : Soit un ensemble de trois (03) personnes R1, R2 et R3 : [Personne : {R1, R2, R3}]. On le déclare comme une collection RDF :

```
< rdf : RDF>
xmlns: rdf=http://''www.w3.org/1999/02/22-rdf-syntaxe-ns#''>
xmlns : gc=''http://www.université/termes#''>
<rdf :Description  rdf :about :'http://www.université/Personne''>
<gc :HasReferent>
<rdf : parseType ='Collection'>
      <rdf :Description  rdf:about=''http://www.université/Personne/R 1''/>
      <rdf :Description  rdf:about=''http://www.université/Personne/R2''/>
      <rdf :Description  rdf:about=''http://www.université/Personne/R3''/>
</gc :HasReferent>
</rdf : RDF>
```

- *Ensemble partiellement spécifié.* Dans ce cas, on ne connaît pas tous les composants de l'ensemble. L'ensemble est composé de l'élément « R1 » et bien d'autres. Dans RDF(S), on déclare un « Container », de type « bag », ayant comme premier élément celui qu'on connaît « R1 ». On sait qu'il n'existe pas de moyen de fermer un container mais, on peut aussi ajouter ici, une URI qui se termine par « _n », où n est un entier différent de zéro.

```
  < rdf : RDF>
    xmlns:rdf=http://''www.w3.org/1999/02/22-rdf-syntaxe-ns#''>
     xmlns : gc=''http://www.université/termes#''>
<rdf :Description  rdf :about :'http://www.Université/Personne''>
<gc :HasReferent>
<rdf : Bag> <rdf :Li     rdf:resource=''http://www.université/ Personne/R1''/>
              <rdf :Li    rdf:resource=''http://www.université/Personne/ _1''/>
</rdf :Bag>
</rdf : Description>
</rdf : RDF>
```

- *Ensemble collectif.* Ces ensembles sont utilisés dans le cas où la même partie du graphe est appliquée à tous les éléments de l'ensemble. RDF permet l'utilisation de containers comme « rdf :Bag ». L'utilisation de l'attribut « rdf :aboutEach » dans l'élément

<rdf :Description> permet ensuite d'affecter des propriétés avec la même valeur à un ensemble de ressources. L'utilisation de l'attribut « BagId » dans l'élément <Description> permet lui de réifier les propriétés contenues dans la déclaration RDF.

Exemple : On veut déclarer trois articles ayant les codes 11, 12 et 13. Ces articles appartiennent tous au même auteur YM. En RDF, on écrit :

```
< rdf : RDF>
xmlns: rdf=http://''www.w3.org/1999/02/22-rdf-syntaxe-ns#''>
xmlns : dc =''http://purl.org/metadata/dublincore-core#''>
<rdf :Description  rdf :about :'http://www.library/Articles/Info.htm''>
 <bagID= ''Classement''>
  <dc :Subject   rdf :resource=''http://www.Articles/info/code# 11''/>
  <dc :Subject   rdf :resource=''http://www.Articles/info/code# 12''/>
  <dc :Subject   rdf :resource=''http://www.Articles/info/code# 13''/>
   </rdf : Description>
  <rdf :Description aboutEach ='Classement'>
        <dc : Creator> YM </dc : Creator>
  </rdf :Description>
  </rdf : RDF>
```

- *Ensembles respectifs.* Chaque élément de l'ensemble respectif entretient une relation avec un élément de l'autre ensemble. En RDF, on peut modéliser cela en spécifiant pour chaque élément, l'élément correspondant par le biais de la propriété représentant la relation entre les deux. En d'autres termes, si un concept [T : resp{R1, R2}] est relié par une relation « Rel » avec un autre concept [T' : resp{R3,R4}], on déclare alors, des relations de type Rel entre chacun des éléments. Et donc, les triplets (Concept1, relation, concept2) sont : (R1, Rel, R3) et (R2, Rel, R4). On déclare alors deux propriétés «Rel» et «Rel1» telles que : «Rel» ayant comme «rdfs :Domain» les deux classes «R3» et «R4», «rdfs :Range» la classe «R1». «Rel1» ayant comme «rdfs :Domain» les deux classes «R3» et «R4», «rdfs :Range» la classes «R2» bien sûr en précisant à chaque fois laquelle des classes est instanciée.

IV.2.4. Relation conceptuelle

Les relations conceptuelles expriment l'existence d'un lien entre des concepts. La notation de relation dans les graphes conceptuels flous peut être interprétée comme suit : « il existe une

relation de *type R.* Elle relie un *concept Source* et un *concept destination* et possède un *nombre de référents* participant à cette relation ». La relation est alors une propriété RDFS ayant comme source une ressource indiquée par «rdfs :Domain» et comme destination une ressource indiquée par «rdfs :range». Le type de la relation peut être déterminée grâce à « rdfs :subPropertyOf ».

On a vu sans le chapitre précédent, qu'il existe des types de relations conceptuelles prédéfinies, comme : 'Agent', 'Expérienceur', 'instrument', …etc. Ces relations peuvent être représentées par des propriétés RDF.

Exemple

- La relation 'Agent' doit avoir comme « rdfs :domain » une classe de type 'Action' et un « rdfs : range » une classe de type « Acteur ».
- La relation de type 'Experienceur' s'applique aux classes de type 'Etat' (rdfs :domain) et prend ses valeurs dans les classes de types 'ANIME'

Cependant, il existe des types de relations 'STAT' et 'STMT' (Voir chapitre 3) qui sont des relations spéciales qui représentent en réalité des graphes emboîtés dont leurs prise en compte se ferra par des descriptions RDF emboîtés dans des conteneurs de types « Bag ». Chaque élément de ce « Bag » est une déclaration RDF exprimé grâce au principe de réification.

Mais, ces déclarations réifiées dans le « bag » sont lourdes à traiter et difficiles à comprendre. Pour cela, la proposition de l'équipe de INRIA au sein de leur projet ACACIA [Sav03] définit une extension de RDFS, appelé DRDFS (Defined Resource Description Framework Schema).

Dans DRDFS, des classes spéciales comme « DefinedClass », « DefinedProperty » et « Context » ainsi que des propriétés comme « isContextOf » sont spécifiées pour exprimés ces connaissances contextuelles.

A ce niveau, il nous reste à déterminer la façon de représenter le nombre de référent. Il est important de noter qu'une relation RDFS (Propriété RDFS) ne peut être que binaire (voir section III). Pour représenter le degré de compatibilité de la relation conceptuelle à son type, elle peut être vue comme une relation n-aire entre à la fois les deux classes conceptuelles et la classe de compatibilité. En d'autres termes, cela peut être lu comme suit : la Classe représentant le concept « C1 » est liée à la classe représentant le concept « C » par une relation R avec un degré de compatibilité « Comp » à son type conceptuel. Pour représenter ce lien, on peut utiliser les nœuds anonymes (sans identifiant précis). Voici le Schéma correspondant :

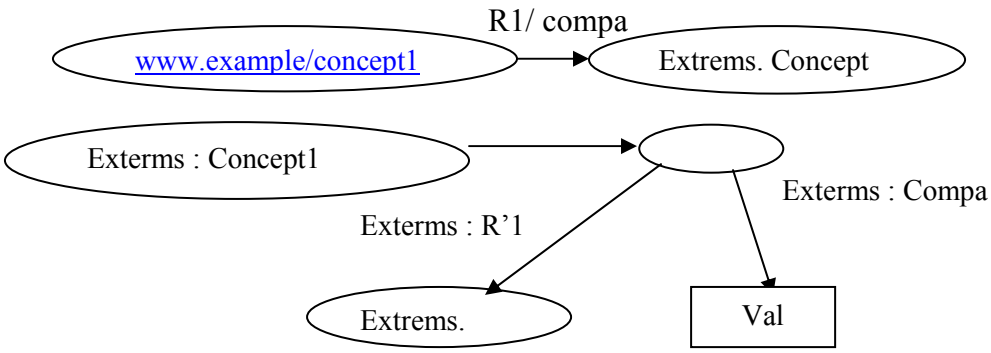

Figure 1. Déclaration RDF avec un nœud anonyme reflétant un concept flou

IV.2.5. Annotations syntaxiques

Les annotations syntaxiques permettent de préserver les nuances syntaxiques une fois que les phrases en langage naturel sont translatées en graphes conceptuels. Pour la faire en RDF (S), l'utilisation de « rdfs : Comment » sera suffisante.

Exemple

Soit le graphe suivant : [Chat : Minou ; n] → (Agnt ; Subj) ←[Etat ; V]

En RDF/ XML on peut l'écrire

```
<rdfs :Class rdf :ID ='Chat'>
        <rdfs:Comment> nom </rdfs:Comment>
</rdfs :Class>
<rdfs :Class rdf :ID ='Etat'>
        <rdfs:Comment> Verbe </rdfs:Comment>
</rdfs :Class>
<rdf :property rdf :ID ='Agnt'>
        <rdfs:domain rdf: resource ='#Chat'>
        <rdfs: range rdf: resource ='#Etat'>
        <rdfs:Comment> Subj </rdfs:Comment>
</rdfs :Class>
```

A ce niveau, la représentation des concepts et des relations conceptuelles, avec prise en compte du flou, peut être faite par aux graphes conceptuels flous. Grâce aux règles de passage proposées leur représentation par des déclarations RDF(S) est devenue possible.

V. Validation par Protégé 2000

Dans le but de valider notre proposition nous avons utilisé l'outil Protégé2000 [11] qui nous permet de définir des classes et des propriétés RDF(S). Rappelons que notre proposition consiste à ajouter une dimension supplémentaire aux différentes ressources RDF représentant la notion du flou dans les graphes conceptuels. Les classes RDF(S) sont aussi des classes dans Protégé2000. Les attributs de ces classes ainsi que les relations entre ces dernières peuvent être représentées grâce à la définition de Slot. Essayons de représenter les deux graphes suivants :

[Homme :Mohamed|vrai]→(Enseigner) →[université :USTHB| vrai]

[Personne :Sara|trèsvrai]→(Donne_TD) →[Faculté:Informatique_Electronique vrai]

Le mapping de ces concepts vers RDFS sera alors : Le concept « homme » est un sous type du concept « Personne ». « Sara » et « Mohamed » sont des instances. Pour représenter la compatibilité, on définit la classe « compatibilité » avec ces trois sous classes : « positive_compatibilité», « négative_compatibilité » et «inconnu_compatibilité » comme dans la figure 2.

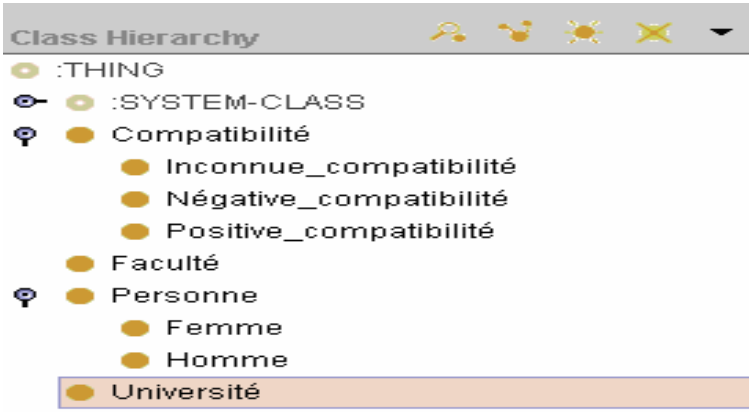

Figure 2. Hiérarchie de classes dans Protégé2000

On créera des slots représentant la valeur de compatibilité de l'instance à son type conceptuel comme pour « Sara » et «Mohamed».

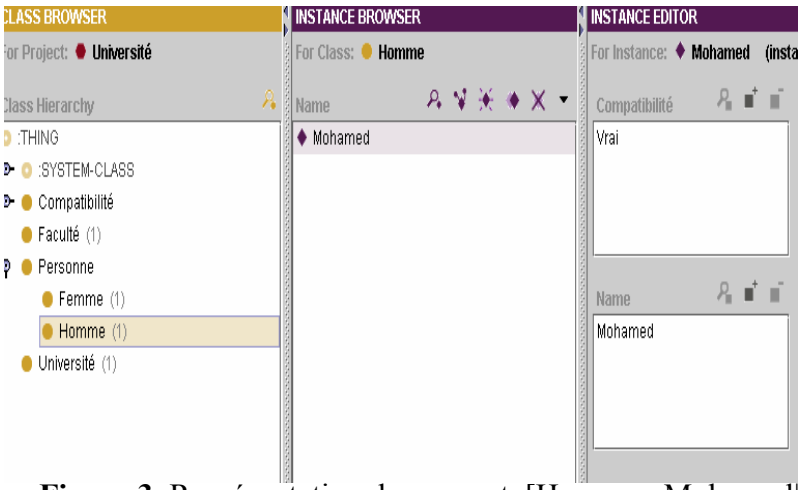

Figure 3. Représentation du concept [Homme : Mohamed|vrai]

Soient les deux relations « Enseigner » et « Donne-TD » où la première relation est le super-type de la seconde relation. Elles sont déclarées, dans Protégé200, grâce à la hiérarchie des Slots (Voir la figure 4).

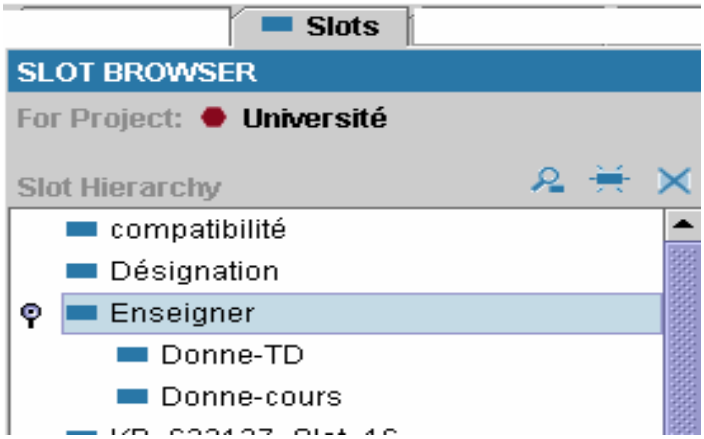

Figure 4. La hiérarchie des relations

Protégé2000 permet de générer le code source associé à ces déclarations grâce à sa propriété d'exporter le projet vers le format RDF Schéma. Le fichier récupéré contient des déclarations RDF(S) qui correspondent aux suppositions des règles de correspondances.

Voici le Code Source récupéré grâce à Protégé2000 :

Code des déclarations des concepts et des relations

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [   <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
      <!ENTITY rdf_ 'http://protege.stanford.edu/rdf'>
      <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>]>
<rdf:RDF xmlns:rdf="&rdf;"
      xmlns:rdf_="&rdf_;"
      xmlns:rdfs="&rdfs;">
<rdf:Property rdf:about="&rdf_;Name"
      rdfs:label="Name">
     <rdfs:domain rdf:resource="&rdf_;Personne"/>
     <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&rdf_;Désignation"
      rdfs:label="Désignation">
     <rdfs:domain rdf:resource="&rdf_;Faculté"/>
     <rdfs:domain rdf:resource="&rdf_;Université"/>
     <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

```
<rdf:Property rdf:about="&rdf_;Enseigner"
      rdfs:label="Enseigner">
     <rdfs:domain rdf:resource="&rdf_;Personne"/>
     <rdfs:range rdf:resource="&rdfs;Université"/>
</rdf:Property>
<rdf:Property rdf:about="&rdf_;Donne-TD"
      rdfs:label="Donne-TD">
     <rdfs:subPropertyOf rdf:resource="&rdf_;Enseigner"/>
     <rdfs:range rdf:resource="&rdf_;Faculté"/>
</rdf:Property>
<rdf:Property rdf:about="&rdf_;Donne-cours"
      rdfs:label="Donne-cours">
     <rdfs:subPropertyOf rdf:resource="&rdf_;Enseigner"/>
     <rdfs:range rdf:resource="&rdf_;Faculté"/>
</rdf:Property>
<rdfs:Class rdf:about="&rdf_;Université"
      rdfs:label="Université">
     <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&rdf_;Faculté"
      rdfs:label="Faculté">
     <rdfs:subClassOf rdf:resource="&rdfs;Université"/>
</rdfs:Class>
<rdfs:Class rdf:about="&rdf_;Femme"
      rdfs:label="Femme">
     <rdfs:subClassOf rdf:resource="&rdf_;Personne"/>
</rdfs:Class>
<rdfs:Class rdf:about="&rdf_;Homme"
      rdfs:label="Homme">
     <rdfs:subClassOf rdf:resource="&rdf_;Personne"/>
</rdfs:Class>
<rdfs:Class rdf:about="&rdf_;Compatibilité"
      rdfs:label="Compatibilité">
     <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&rdf_;Inconnue_compatibilité"
      rdfs:label="Inconnue_compatibilité">
     <rdfs:subClassOf rdf:resource="&rdf_;Compatibilité"/>
</rdfs:Class>
<rdfs:Class rdf:about="&rdf_;Négative_compatibilité"
      rdfs:label="Négative_compatibilité">
```

```
    <rdfs:subClassOf rdf:resource="&rdf_;Compatibilité"/>
</rdfs:Class>
<rdfs:Class rdf:about="&rdf_;Personne"
     rdfs:label="Personne">
    <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&rdf_;Positive_compatibilité"
     rdfs:label="Positive_compatibilité">
    <rdfs:subClassOf rdf:resource="&rdf_;Compatibilité"/>
</rdfs:Class>
<rdf:Property rdf:about="&rdf_;compatibilité"
     rdfs:label="compatibilité">
    <rdfs:domain rdf:resource="&rdf_;Compatibilité"/>
    <rdfs:domain rdf:resource="&rdf_;Faculté"/>
    <rdfs:domain rdf:resource="&rdf_;Personne"/>
    <rdfs:domain rdf:resource="&rdf_;Université"/>
    <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
</rdf:RDF>
```

Code de déclarations des référents (instances)

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
                                        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
                                        <!ENTITY université 'http://protege.stanford.edu/system#'>
                                        <!ENTITY rdf_ 'http://protege.stanford.edu/rdf'>
                                        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>]>
<rdf:RDF xmlns:rdf="&rdf;"
          xmlns:rdf_="&rdf_;"
          xmlns: university="&université ;"
          xmlns:rdfs="&rdfs;">
<rdf_:Femme rdf:about="&rdf_;KB_633137_Instance_21"
        rdf_:Name="Sara"
        rdf_:compatibilité="très_faux"
        rdfs:label="Sara"/>
<rdf_:Homme rdf:about="&rdf_;KB_633137_Instance_22"
       rdf_:Name="Mohamed"
       rdf_:compatibilité="Vrai"
        rdfs:label="Mohamed"/>
<rdf_:Université rdf:about="&rdf_;KB_633137_Instance_23"
```

```
        rdf_:Désignation="USTHB"
        rdf_:compatibilité="Très_vrai"/>
<rdf_:Faculté rdf:about="&rdf_;KB_633137_Instance_24"
        rdf_:compatibilité="Vrai"
       <rdf_:Désignation xml:space='preserve'><![CDATA[INFORMATIQUE & ELECTRONIQUE]]>
        </rdf_:Désignation>
</rdf_:Faculté>
<université :_instance_annotation rdf:about="&rdf_;KB_633137_Instance_25"
 université :_creation_timestamp="2006.04.20 00:26:08.922 CET"
 université :_creator="Nawel"
 rdfs:label="KB_633137_Instance_25"/>
</rdf:RDF>
```

VI. Exemple d'application

La prise en compte de la multi- expertise dans les projets d'intelligence artificielle a montré l'insuffisance et les limites des méthodes et outils classiques pour la gestion des conflits lors de la phase d'acquisition de connaissances à partir des multiples experts.

Lorsque le développement d'un système à base de connaissances implique plusieurs sources d'expertises (experts), le cogniticien (ingénieur de connaissances) rencontre plusieurs types de conflits. En effet, les experts en raison de leurs différences de formation par exemple, peuvent utiliser des terminologies ou des méthodes différentes pour résoudre le même problème.

C'est dans ce contexte, que nous avons utilisé une application utilisant le processus qui est une méthode coopérative de résolution. L'outil développé pour soutenir le processus est fortement interactif. Il est conçu pour provoquer la discussion de la résolution des conflits par la négociation entre des experts [Akl05] autant que mettre à jour la résolution appropriée.

L'outil a pour but de guider les experts dans leur négociation jusqu'à ce qu'ils atteignent une résolution finale qui les satisferont. Le formalisme utilisé étant les graphes conceptuels pour une application permettant d'établir un diagramme de transition d'état pour les livres d'une bibliothèque.

Chacun des experts utilise un nombre d'interface afin d'introduire les états et les relations de son graphe conceptuel [Figure 5]

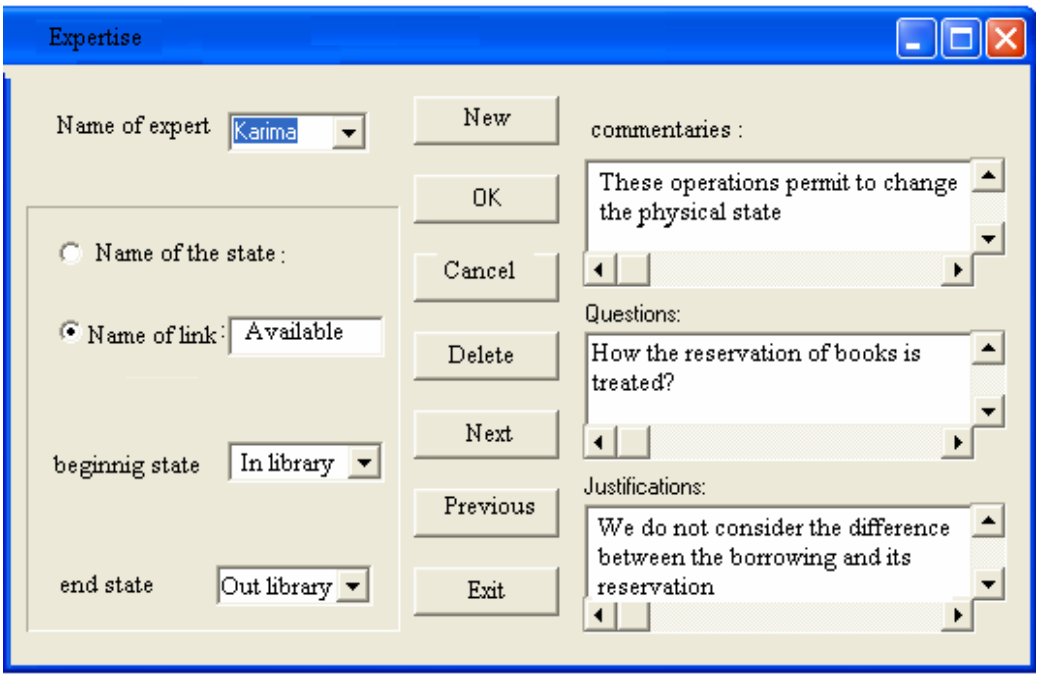

Figure 5. Introduction des concepts et des relations conceptuelles

Chaque expert aboutira à un graphe conceptuel [Voir figures 6, 7] correspondant à deux points de vue différents

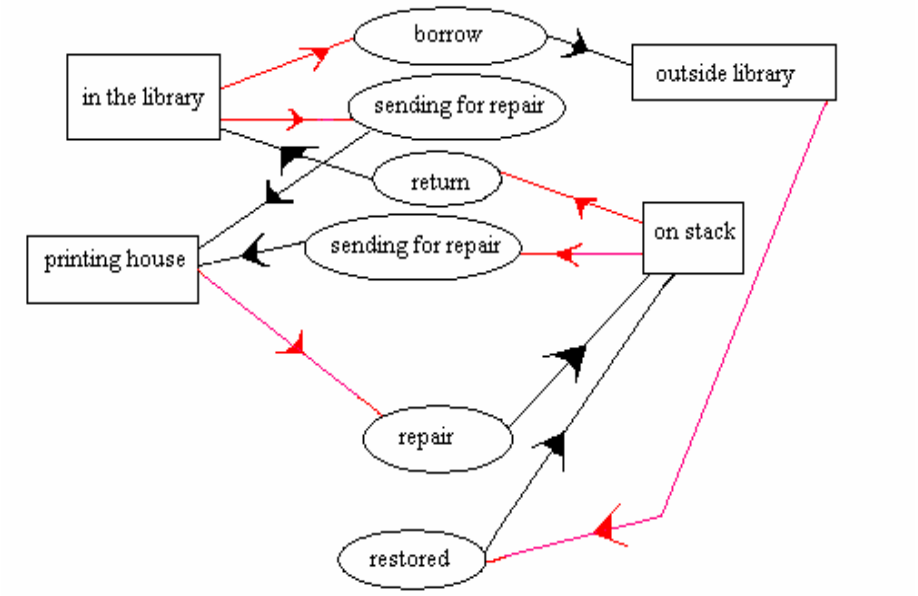

Figure 6 . Graphe conceptuel du point de vue de la localisation des livres

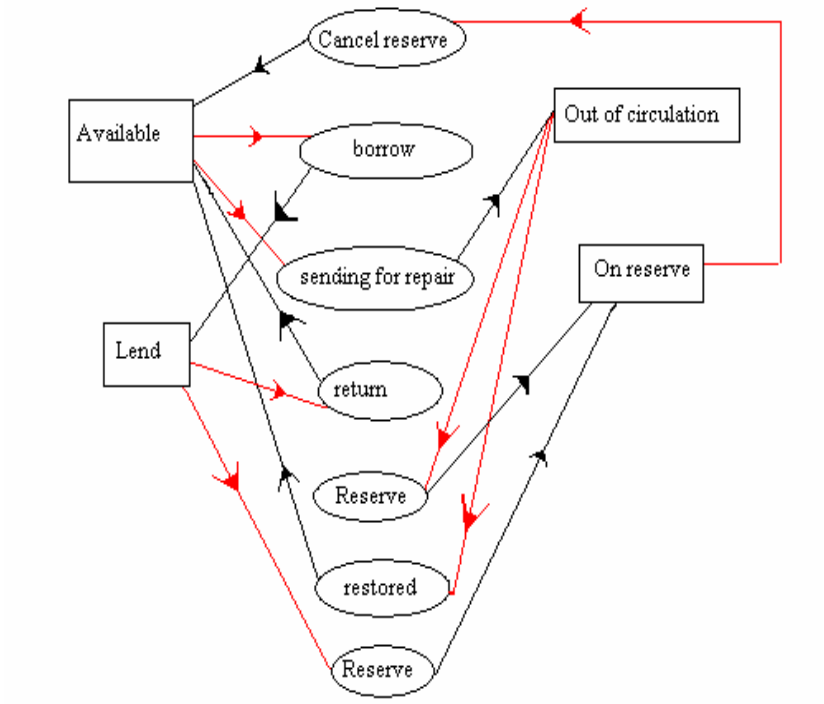

Figure 7. Graphe conceptuel du point de vue de disponibilité des livres

L'outil détecte automatiquement les correspondantes et les conflits et après le déclenchement de la phase de négociation, un graphe commun (Figure 8) est obtenu résultant d'un consensus entre les experts. Des options de résolution non consensuelles ne sont pas prises en compte dans cette description

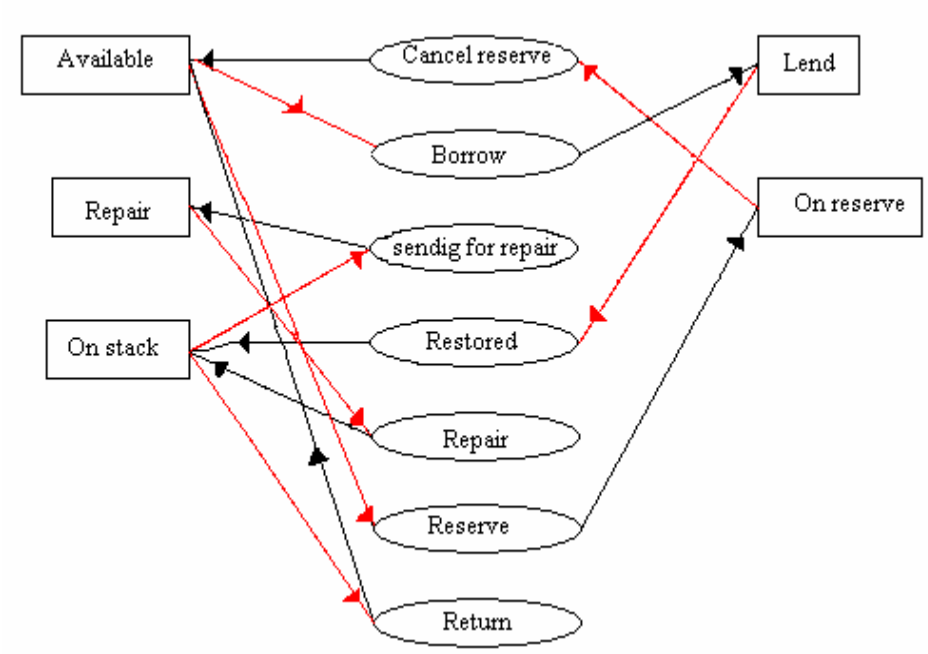

Figure 8. Graphe conceptuel commun obtenu après négociation

L'auteur a souligné dans [Akl05] que les concepts utilisés n'étaient pas toujours précis ou certains. En effet, les connaissances représentent les expressions de la langue naturelle peuvent

parfois être floues, de nature vague, imprécise, incertaine, etc. Pour cela il a mentionné dans ses travaux futurs d'étendre ce processus de négociation en utilisant des graphes conceptuels flous au lieu des graphes conceptuels classiques.

En plus, il serait intéressant de réaliser ce processus entre plusieurs experts à travers un réseau à distance ou pourquoi pas à travers le Web. Pour cela il a utilisé dans un deuxième travail qui est maintenant en cours d'élaboration les règles de passages qu'on a proposé pour étendre le logiciel vers le web sémantique.

Le logiciel englobe les interfaces déjà présentées jusqu'ici en ajoutant bien sur une valeur floue dans la phase de définition de concepts et de relations. Une interface principale propose à l'expert l'introduction de ses concepts et ses relations. Les deux types de concepts peuvent être introduits suivants que le concept est un concept d'entité flou ou un concept d'attribut flou.
La définition de concept se fait en précisant son type, son degré de compatibilité (Voir figure 9) ainsi que les relations qui peuvent le relier avec les autres concepts (Voir figure 10).

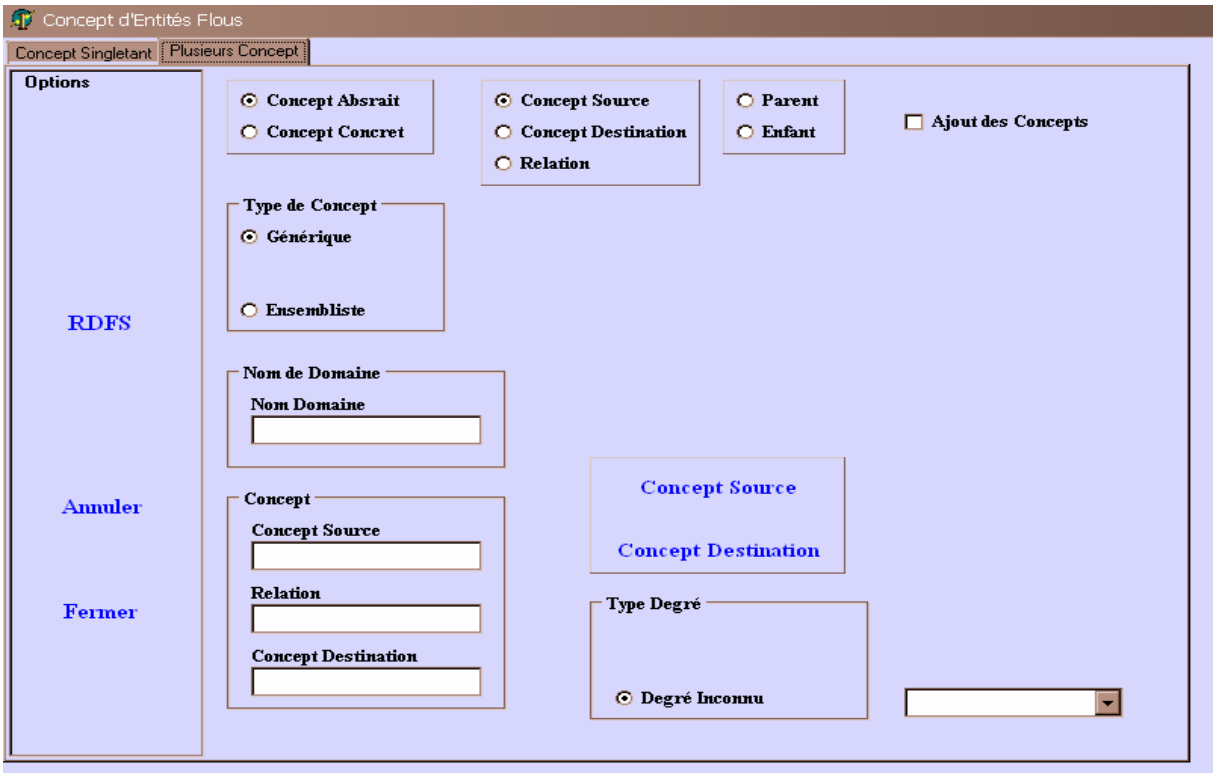

Figure 9. Définition des concepts et de relation conceptuelle pour des concepts à référents ensemblistes

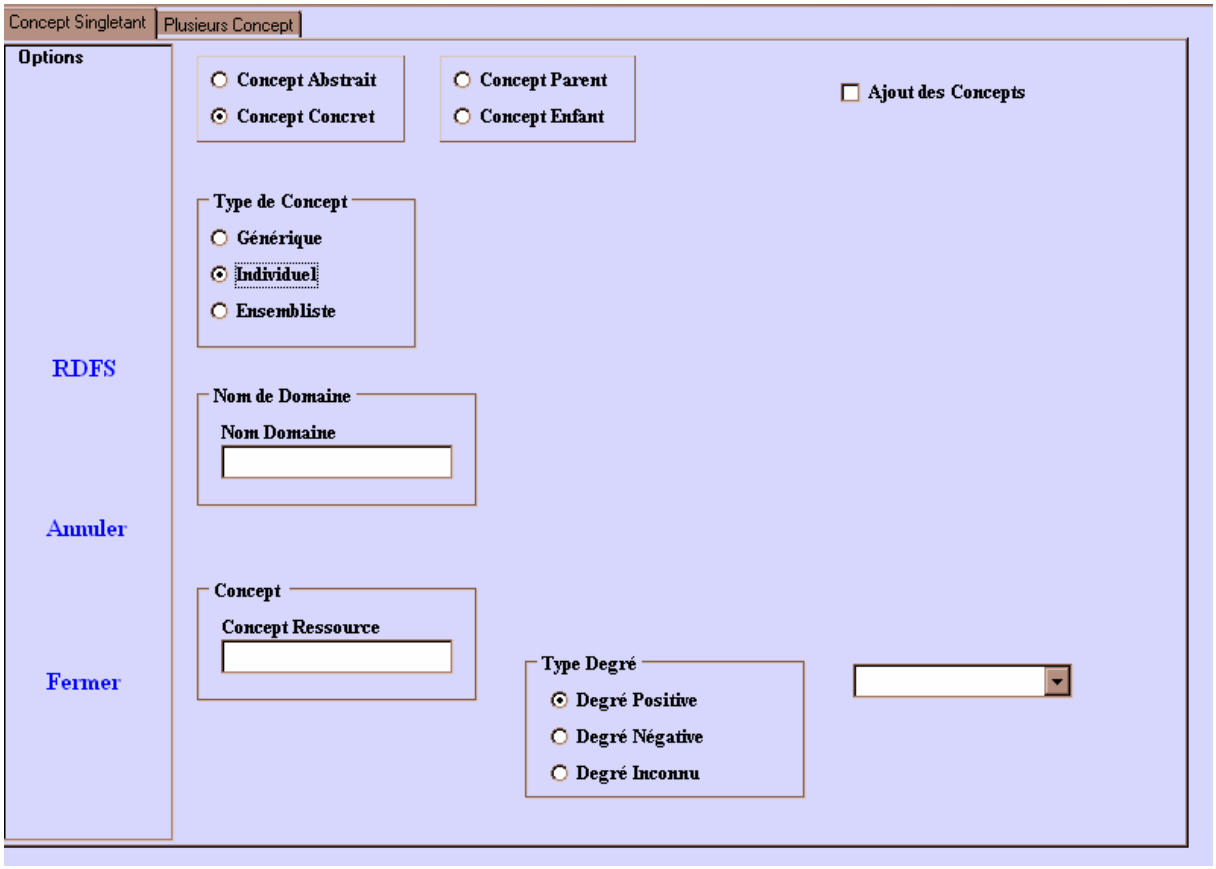

Figure 10. Définition des concepts génériques

Une fois la négociation terminée, le passage du graphe conceptuel résultant vers les déclarations de classes et de propriétés RDF (S) est faite grâce au bouton [RDFS].

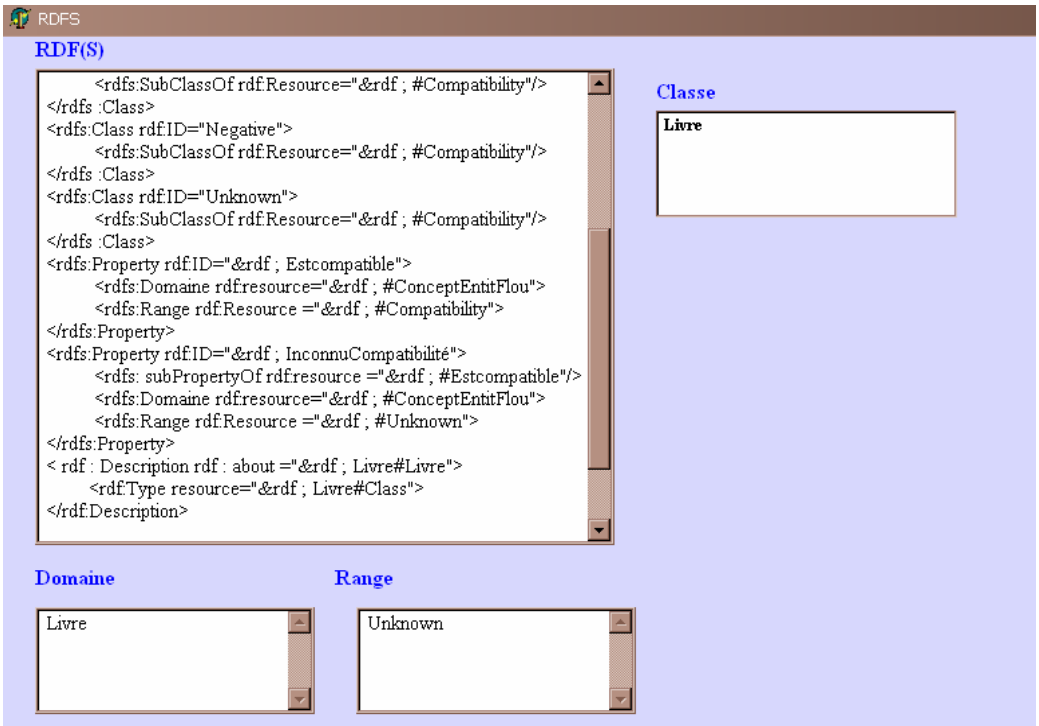

Figure 11. Déclaration RDF(S) correspondante à une définition de concept

VII. Conclusion

Il est clair à ce niveau, que si les ressemblances entre les deux modèles de représentation sont bien exploitées, un passage de chacun des modèles vers l'autre est facile à mettre en place.

La validation par Protégé2000 est l'une des solutions qui nous a permis de savoir que si les règles appliquées donnent des résultats acceptables.

L'application, qui est toujours en phase de développement, permettra à son tour aux experts d'exploiter le processus de négociation à travers un web en prenant en compte l'aspect sémantique.

Conclusion générale

Conclusion Générale

Notre travail, d'un point de vue architectural, intervient dans la première couche du web sémantique à savoir *les métadonnées*. Ces dernières permettent d'assurer l'interopérabilité entre les différents formalismes et ontologies.

Nous sommes partis d'une application développée dans le cadre d'un projet de l'équipe « représentation de connaissances » au sein du laboratoire LRIA. Il s'agit de la spécification d'un système de librairie où la connaissance de l'état (disponible, en réparation, exclu du prêt, etc…) d'un livre (ressource physique) est décrite par le biais de *métadonnées* modélisées par le formalisme des graphes conceptuels. Nous avons considéré les concepts flous dans la représentation des graphes conceptuels. Puis, nous nous sommes intéressés à RDF(S) langage d'annotations et de métadonnées afin de réutiliser l'application dans le cadre du web sémantique.

Nous avons montré les points communs et les différences entre les deux modèles de représentation puis un ensemble de règles a été extrait permettant le passage entre les deux formalismes.

Ces règles ont été validées par l'éditeur d'ontologies Protégé 2000 [Pro00]. Actuellement, un outil en cours de développement dans le cadre d'un projet de fin d'étude, réalise automatiquement une partie de ses règles.

Des perspectives sont à envisager :

- Des langages de requêtes tels que RDQL [RDQ04], SPARQL [SPA06], peuvent être employés pour répondre à des requêtes utilisateurs. Une étude plus approfondie de ces langages ou même une proposition d'un langage de requête plus adéquat fera l'objet de travaux futurs.

- RDF et RDF(S) ne permettent pas d'introduire une normalisation sur la sémantique. Aussi, plusieurs utilisateurs peuvent donner des noms différents à des entités de même sens, ce qui empêchera l'agrégation entre des données provenant de sources web différentes. La solution viendrait du développement *d'ontologies orientées concepts* qui traite des concepts

fondamentaux du monde cible et qui demande d'être examinés en profondeur. Ceci est réalisé par l'utilisation du langage ontologique OWL (Ontology Web Language) [OWL04]. Ce dernier permet de publier et de partager des ontologies sur le web.

- La prise en compte des différentes définitions d'un concept ainsi qu'une structuration différente de la connaissance fera l'objet de la suite de notre travail. Notre objectif futur étant la construction d'une ontologie avec prise en compte de points de vues mutuellement exclusifs.

Références

Références

[Akl05] K. Akli- Astouati, « Assistance Tool for Conflict resolution by technical negotiation», in the first International symposium on information & communication Technologies (ICTIS'05), Tetuan, Morocco, pp 188 – 194, Essaidi, M. Raissouni,N. (eds), 2005.

[Ber01] T., Berners- Lee, J., Hendler, O., Lassila, «The Semantic Web», Scientific American, 2001.

[Bou95] B., Bouchon – Meunier, « La logique flou et ses applications », Edition Addison – Wesley, 1995.

[Bun85] H. Bunt, «Mass terms and Model – Theoretic Semantics» Cambridge University Press, Cambridge, 1985.

[Cao96] T.H. Cao, W., Wuwongse, «Towards Fuzzy Conceptual Graph Programs». In proceedings of the 4th International Conference on Conceptual Structures, ICCS' 96. Lecture Notes in Artificial Intelligence, Volume 1115, page 263 – 276, Sydney, Australia, Springer–Verlag, 1996.

[Cao01] T.,H., Cao, «Fuzzy Conceptual graphs for the Semantic Web» invited to the Berkley Initiative in Soft computing International Workshop on Fuzzy Logic and the Internet, 2001.

[Cha00] J., Charlet, M.Zacklad, G. Kassel «Engagement Sémantique et Engagement ontologique », dans « *Ingénierie de connaissances, évolutions récentes et nouveaux défis* » Eyrolles, 2000.

[Cha02] J., Charlet, « Ontologie », Semaine ISDN/ Web Sémantique, STIM/ DSI/ AP- HP & Université de Paris 6 & groupe « Terminologie et IA », 24 Mai 2002.

[Cha03] J., Charlet, B., Bachimont et R., Troncy, «Ontologie pour le Web Sémantique», Action Scientifique CNRS/ STIC Web Sémantique – Rapport Final V2, 13 Oct 2003

[Che99] W. Chen and R. Mizoguchi. "Communication Content Ontology For Learner Model Agent in multi-Agent Architecture". Workshop on Anthologies for Intelligent Educational Systems, Ninth International Conference on Artificial Intelligence in Education, AI-ED'99, Le Mans, France, July 19- 23th, 1999. http://www.ei.sanken.osaka-u.ac.jp/aied99/aied99-onto.html

[Del03] A., Delteil, C., Faron et R., Dieng, «Le modèle des graphes conceptuels pour le web sémantique. Extensions de RDF et RDFS basées sur le modèle des graphes conceptuels». RSTI – l'Objet – 9/2003. XML et les objets, 2003.

[Eas91] S. Easterbook, « Handling conflict between domain descriptions with computer-Supported negotiation », Knowledge Acquisition Vol3, September pp 255- 289, 1991.

[Fen98] D. Fensel, S. Decker, M. Erdmann, and R. Studer, "Ontobroker: How tomake the WWW Intelligent". Research report, Institute AIFB, in Proceedings KAW98, the 11th Banff Knowledge Acquisition for Knowledge-Based System Workshopresearch report, Banff, Canada, April 1998.

[Fur02] F.Fûrst, «L'ingénierie Ontologique – Ingénierie de Connaissances », Rapport de recherche N° 07/02, Institut de recherche en informatique de Nantes, 2002.

[Gar98] Guarino, N. "Ontology and Information Systems" In N. Guarino (ed), Formal Ontology in information system. Proc of the 1st International Conference, Trento, Italy, 6 - 8 June 1998. IOS Press

[Gar99] Guarino, N., "Ontoseek, Content–based Access to the web", IEEE intelligent System, 1999.

[Goo72] N. Goodman, «Problems and projects», Bobbs. Merrill Co.New York, 1972.

[Gru91] Thomas R. Gruber. "The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases". In Proceedings of the Second International Conference Principles of Knowledge Representation and Reasoning, (KR & R-91), J. Allen, R. Fikes, and E. Sandewall (Eds.), Morgan Kaufmann Publishers: San Mateo, CA, pp.601-602, 1991.

[Gru93] T.R. Gruber. "A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition", Vol.5, No. 2, pp.199-220, 1993.

[Gru95] T.R. Gruber. “Toward Principles for the Design of Ontologies Used for KnowledgeSharing”. Revision of paper presented at the international workshop on Formal Ontology, Padova, Italy,March 1993, in Special issue of the International Journal of Human-Computer Studies, Vol. 43, N° 5-6, Nicola Guarino and Roberto Poli (Eds.), 1995.

[Grü95] M. Grüninger and Mark S. Fox. “Methodology for the Design and Evaluation of Ontologies”. Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI, Montreal, 1995.

[Hac03] M.S. Hacid, C. Reynaud, «Intégration de Sources de données» Action Scientifique CNRS/ STIC Web Sémantique – Rapport Final V2, 13 Oct 2003.

[Hea04] O., Haemmerlé, «Des graphes conceptuels pour la représentation de connaissances en micro – biologie» Thèse de Doctorat, INRIA Sophia Antipolis, 2004.

[Ike99] M. Ikeda, Y. Hayashi, J. Lai, W. Chen, J. Bourdeau, K. Seta and R. Mizoguchi. “An ontology more than a shared vocabulary”. Workshop on Ontologies for Intelligent Educational Systems, Ninth International Conference on Artificial Intelligence in Education, AI-ED’99, Le Mans, France, July 19-23,1999.

[Kle00] M. Klein, D. Fensel, F.van Harmelen and I. Horrocks. “*The Relation between Ontologies and Schema-Languages: Translating OIL-Specifications to XML-Schema”*. In Proceedings of the Workshop on Applications of Ontologies and Problem-solving Methods, 14th European Conference on Artificial Intelligence ECAI-00, Berlin, Germany, August 20-25th 2000.
http://www.ontoknowledge.org/oil/papers.shtml

[Lau03] P., Laubet, J. Charlet et C., Raynaud, «Introduction au Web Sémantique», Journée «Web sémantique et Sciences humaines et sociales » WS-SHS-2003, «Web Sémantique», CNRS/STIC 07 mai 2003.

[Lus62] C.S. Luschei, «The logical systems of Lesnieski» North – Holland Publishing CO, Amsterdam, 1962.

[Mar95] P. Martin. “Using the WordNet Concept Catalog and a Relation Hierarchy for Knowledge Acquisition”. Proc. of Peirce'95, 4th, International Workshop on Peirce, University of California, Santa Cruz, USA, pp. 36-47, August 18th 1995.

[Mic99] A. Michard, « XML Langage and Applications », Eyrolles, 1999.

[Miz04] R. Mizoguchi, « le rôle de l’ingénierie ontologique dans le domaine des EIAH » Sticef.org, Revue Science et Technologies de l’information et de la communication pour l’Education et de la Formation, Vol 11, 2004.

[Mor87] S.K. Morton, “Conceptual Graphs and fuzziness in artificial Intelligence” Thèse de Doctorat, Université de Bristol, 1987.

[Mug96] M.L., Mugnier, M. Chein, «Représenter des connaissances et raisonner avec les graphes», Revue d’intelligence Artificielle, volume 10 N° 01 page 7 -56, 1996.

[Pei82] C.S.Peirce. Lettre, Peirce to O.H. Mitchell. In Writtings of Cherles, S.Peirce, Vol 4, pp 394- 399, MA, Indiana University Press, Bloomington IR, 1982

[Pri03] Y., Prie, S. Garlati, “Métadonnées et annotations dans le web Sémantique”, Action Scientifique CNRS / STIC, Web Sémantique – Rapport Final, 2003.

[Sav03] O., Savoire, «Corese ODL», Activity Report, ACACIA Projet, www.inria/acacia, , June 2002- May 2003.

[Sch97] J.R. Schoening. “A Case and Strategy for Developing Standardized Educational Domain Modules and Ontologies”. P1484 Working and Study Developing Technical Standard for Learning Technology, 1997.

[Sow84] J.F. Sowa, “Conceptual Structures “, Addison Wesley, 1984.

[Sow88] J.F. Sowa, “Lexical structures and conceptual structures- in semantics in the lexicon”, J., Pustejovesley (ed), 1988.

[Sow91] J.F. Sowa, “Toward the Expressive Power of natural Language” In Principles of Semantic Networks – Explorations in the representation of knowledge, pp 177- 182. Edited by John F. Sowa, IBL systems Research, 1991.

[Wuw93] V. Wuwongse, «Fuzzy Conceptual Graphs» In proceeding of the 1st International Conference on Conceptual Structures, ICCS'93, Lecture Notes in Artificial Intelligence n° 699, pages 430-449, Quebec City, Canada, Springer-Verlag, August 1993..

[Zad65] L., Zadeh, «Fuzzy Set», Information and control, Vol 8, page 338 – 353, 1965.

Sites web

[Dub00] Dublin Core Metadata (CR Meeting Helsinki)

[Hay04] P. Hayes, « RDF Semantics » W3C recommendation, World Wide Web Consortium, February 10, 2004 http://www.w3c.org/TR/rdf-testcases/ .

[Hel02] J. Helpin, R., Raphaël J., Volz Dale, 2002, «Besoin pour un langage d'ontologie du Net». www.w3.org/TR/2002/WD-WebOnto-reg-20020307

[Rdf 04] F. Manda, E. Miller, 2004, «RDF Primer» www.w3.org/TR/2004/REC-RDF-Primer-20040210

[OWL04] "OWL Web Ontology Overview" http://www.w3.org/TR/2004/REC-owl-features-20040210/

[Ont98] OntoBroker Project http://ontoBroker.aifb.uni-Karlsruhe.de/index_ob_html

[Pro00] Protegé2000 http://protege.stanford.edu/

[RDQ04] A. Seaborne, "A Query Language for RDF", 9 January 2004 http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/

[SPA06] E. Prud'hommeaux, "SPARQL RDF Query Language", 6 April 2006 http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/