Université de Montréal

Alignement des ontologies OWL-Lite

par
Mohamed TOUZANI
Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maîtrise ès sciences (M.Sc.)
en informatique

Avril, 2005

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

Alignement des ontologies OWL-Lite

présenté par :
Mohamed TOUZANI

a été évalué par un jury composé des personnes suivantes :

Esma Aïmeur
(présidente-rapporteure)
Petko Valtchev
(directeur de recherche)
Balázs Kégl
(membre du jury)

Mémoire accepté le :
03 octobre 2005

Sommaire

Les ontologies sont devenues un moyen standard de description des ressources sur le Web [Chandrasekaran et al., 1999], offrant la possibilité de partager, et de réutiliser ces ressources à l'échelle du Web. En outre, les ontologies ont ouvert la voie vers la construction des services intelligents sur le Web. Toutefois, si les ontologies constituent un moyen efficace pour pallier à l'hétérogénéité des ressources sur le Web, leur utilisation manifeste à son tour la même problématique [Doan et al., 2001, Noy et McGuinness, 2001, Gruber, 1995].

L'alignement d'ontologies [Euzenat et Valtchev, 2003, Doan et al., 2001] représente la solution ultime pour pallier au problème d'hétérogénéité d'ontologies sur le Web. L'alignement consiste à chercher parmi les couples d'ontologies à aligner, ceux les plus semblables.
Il est évident que l'alignement manuel d'ontologies est un processus long, fastidieux même infaisable. On outre, il heurte considérablement à l'objectif de construire un Web sémantique doté d'outils intelligents. Ainsi, les recherches sur les méthodes d'alignement automatiques sont nombreuses et diverses [Melnik, 2002, Doan et al., 2003, Doan et al., 2002, Sekine et al., 1999]. En effet les méthodes proposées font recours à des techniques très variées souvent combinées. Cependant, elles n'exploitent que partiellement l'information que peut contenir l'ontologie.

Par ce présent travail, nous présentons une méthode d'alignement d'ontologies décrites dans le langage OWL-Lite. La méthode tire avantage de toutes les caractéristiques du langage OWL-Lite -en vois de devenir un standard-. En effet, elle définit une mesure de similarité par catégorie d'entités exploitant toute l'information descriptive : structurelle, contextuelle, les types de données etc.. des entités [Euzenat et Valtchev, 2004] pour un calcul plus précis de la similarité. La méthode fait aussi face au problème de circularité qui empêche le calcul direct de la similarité et fait preuve d'un degré important d'automaticité.

L'implémentation de la méthode d'alignement a donné naissance au système OLA qui offre des fonctionalités de visualisation, de production et de comparaison d'alignements d'ontologie OWL-Lite. Avec OLA nous avons eu l'occasion de participer à la compétition sur l'alignement d'ontologies EON dans le cadre du Workshop sur l'alignement d'ontologies sur le Web.

Mot clés : WEB SÉMANTIQUE, ONTOLOGIE, HÉTÉROGÉNÉITÉ, ALIGNEMENT, OWL, OLA.

Abstract

With the tremendous growth of the Semantic Web, the use of ontologies has become the standard way to describe, share and reutilize resources across the Web [Euzenat et Valtchev, 2004]. However, one of the stringent problems that hampers the use of ontologies is heterogeneity [Noy et McGuinness, 2001, Gruber, 1995]. In fact, ontologies are not necessary described in the same formalisms and languages. Hence, to transform the semantic Web to a more collaborative arena, it relies unquestionably on finding tools to insure reconciliation between ontologies.

Ontology alignment [Euzenat et Valtchev, 2003, Doan et al., 2002] represents an efficient solution to tackle the heterogeneity problem . It can be described as the process of finding relationships between entities (classes, properties, individuals, relations ...). Nowadays numerous tools using different technics -more likely combined technics- to approach the alignment process [Melnik, 2002, Doan et al., 2003, Doan et al., 2002, Sekine et al., 1999].

This thesis capitalized on researches conducted jointly by the Montréal university and INRIA Institute teams. The initiative has come up with a new method that relies on similarity based approach for OWL-Lite ontologies alignment. The method is proved to be powerful and flexible enough to match the expressive power of the OWL-Lite language. It defines an universal measure for comparing entities taking advantage of all the information encompassed by the ontology entities description namely, structural, contextual and data types. Its also overcomes efficiently the circularity problem that hinders the direct similarity computation.

Finally, the implementation of this work has lead to OLA : an open source system offering some valuable functionalities for aligning, visualizing and manipulating the OWL-Lite ontologies. The beta version of OLA is a first step towards a complete alignment system addressing OWL-DL and OWL-Full languages [Euzenat et al., 2004b].

Key words : SEMANTIC WEB, ONTOLOGY, HETEROGENEITY, ALIGNMENT, OWL, OLA.

Table des matières

Table des figures

Liste des sigles et abréviations

ACRONYME	DESCRIPTION
DARPA	The Defense Advanced Research Projects Agency
DAML	DARPA Agent Modeling Language
DAML+OIL	Ontology Lite Alignment
DARPA	The Defense Advanced Research Projects Agency
DTD	Document Type Definition
FCA	Formal Concept Analysis
JWNL	JAVA WordNet Librairy
LD	Les Logiques de Descriptions
P2P	Peer to Peer
OLA	Ontology Lite Alignment
OWL	Ontology Web Language
RDF	Resource Definition FrameWork
RDFSchema	Resource Definition FrameWork Schema
UML	Unified Modeling Language
URI	Unified Resource Identifier
XML	eXtensible Markup Language

Remerciements

Je voudrais exprimer ma sincère gratitude à mon directeur de recherche, Petko Valtchev, de m'avoir proposé et financé ce travail. C'est grâce à son dynamisme et à sa collaboration que ce travail a pu voir le jour.

Je tiens à remercier également, Jérôme Euzenat, professeur à l'INRIA Alpes France, dont la patication a été de grand apport pour la réalisation de ce travail.

Je remercie les membres du jury, Esma Aïmeur et Kégl Balázs d'avoir pris le temps de lire ce mémoire et de me faire part de leurs remarques.

Toute la gratitude et la reconnaissance vont à mes parents pour leur encouragement et leur soutien le long de mon parcours, que Allah les récompense.

Un grand merci pour mes frères et sœurs, en premier lieu Ahmed et Bouchra.

Je tiens à ne pas oublier de remercier mes collègues au laboratoire GELO, nouvellement nommé GEODES, Samah, Amine et Kamal. Je souhaite en particulier bon courage à ceux qui commenceront bientôt la rédaction de leurs thèses.

Je tiens aussi à remercier mes amis Jawad, Doha, Sine, khadija, Hamid, Samia, Hassan, Salwa, Rachid, Alae, Abdelwahid, Saad, Mustapha, Sanae de leur support.

Acknowledgement

First of all I'm grateful to the grace and the help of Allah the Creator of the the universe, who says in Sourat Alalaq :

> ***Read in the name of (in a spirit of respect for) your Guardian/Lord who created. Created mankind from a clinging thing (a clot of congealed blood). Read ! And thy Guardian/Allah is Most Generous. The one who taught by the Pen. Taught man what he was not previously knowing.***

The first person I'm grateful to is my supervisor, Petko Valtchev for his patience, encouragement, and his financial support. I'm deeply entitled to his thoroughly monitoring witch has brought this project to success.

I'd like also to thank Jérome Euzenat who has been working jointly and actively with us.

I'd like also to express my appreciation to the members of jury namely Esma Aïmeur and Balázs Kégl for they spared the time to read this thesis, and get the best out of it.

My deep and sincere thanks to my parents for their long-distance support and for their prayers. This work is yours.

My warm thanks are to my sisters and my brothers for their niceness, and for their brotherhood.

I wish also to thank my colleagues Samah, Hanine, Amine and Kamal. It's was a pleasure to have your company.

At last not at least, I'd like to thank my friends Jawad, Doha, Sine, khadija, Hamid, Samia, Hassan, Salwa, Rachid, Alae, Abdelwahid, Saad, Mustapha and Sanae.

Chapitre 1

Introduction et problématique

Le progrès du Web et toutes les technologies afférentes ont pavé la voie pour la constitution du Web sémantique. Ce dernier est envisagé comme une extension du premier, qui fournit plus qu'un simple accès à l'information, des services s'apparentant au raisonnement. Pour ce faire, le Web sémantique va s'appuyer sur une infrastructure de documents dont le contenu est décrit d'une manière structurée à travers des langages basés sur RDF et XML [Baader et al., 2003, Decker et al., 2000] et donc exploitable par des mécanismes "intelligents" de recherche, de transformation, et d'appariement, etc.

Tout comme le Web "syntaxique", le Web sémantique sera composé de ressources distribuées et hétérogènes qu'il faudrait faire collaborer [Bach et al., 2004]. Pour palier à cette hétérogénéité, un effort considérable sera consacré à produire les moyens effectifs d'assurer l'inter-opérabilité entre ces ressources. Un premier pas dans cette direction est l'utilisation d'ontologies [Chandrasekaran et al., 1999] comme des préférentiels dans la description de ressources. L'approche basée sur les ontologies s'est avérée fructueuse, mais n'a pas atteint une complète inter-opérabilité car l'existence de plusieurs ontologies sur un même domaine - un fait aujourd'hui - est en soi une source d'hétérogénéité. La réponse à cette nouvelle difficulté semble se situer dans le développement des méthodes de réconciliation entre ontologies [Euzenat et Valtchev, 2004, Magini et al., 2003, Doan et al., 2001], ou, pour employer le terme technique, d'alignement d'ontologies.

D'une manière générale, l'alignement [Noy et Musen, 2001, Euzenat et al., 2004a] consiste, étant donnée deux ontologies indépendantes, couvrant la même réalité totalement ou partiellement, à produire un ensemble de couples de concepts des ontologies respectives. Les couples sont interprétés comme des appariements entre les concepts allant d'une similitude jusqu'à l'équivalence parfaite entre les deux concepts. Divers domaines limitrophes à l'ingénierie d'ontologies ont produit des procédés algorith-

miques [Madhavan et al., 2001, Stumme et Maedche, 2001, Euzenat et al., 2004b]qui peuvent servir de base pour le développement de méthodes effectives d'alignement. A noter en particulier la recherche sur la modélisation conceptuelle en bases de données ou en génie logiciel. Cependant, les algorithmes existants ne couvrent qu'une partie des aspects qui peuvent être exprimés dans une ontologie. Le défi majeur aujourd'hui est donc de proposer une méthode qui soit la plus complète sur le plan de l'expressivité du langage de description pris en compte.

Avec le progrés des recherches menées par le World Web Consortium (W3C) sur le langage de description d'ontologies sur le Web OWL -en voie de devenir un standard-, le besoin à des méthodes d'alignement d'ontologies décrites en ce langage est de plus en plus ressenti. Dans cette vision, nous avons proposé une méthode d'alignement d'ontologies OWL-Lite sur le Web. La proposition s'est concrétisée par la mise en œuvre d'un environnement OLA [Euzenat et al., 2004b] d'alignement, de manipulation et de visualisation d'ontologies OWL-Lite.

1.1 Problématique

La description de ressources sur le Web sous forme d'ontologies représente une solution inéluctable au problème d'hétérogénéité. Cependant, les ontologies manifestent à leur tour la même problématique. Ainsi, l'hétérogénéité d'ontologies freine aussi bien leur utilisation universelle que le développement de technologies autour du Web sémantique.
En plus des considérations techniques, l'hétérogénéité d'ontologies trouve sa justification dans beaucoup de facteurs, parfois conflictuels, ce qui suit quelques un les plus importants :

- Multiplicité d'ontologies : le même domaine d'intérêt est décrit par plusieurs ontologies.
- Chevauchement d'ontologies : une conséquence directe de la multiplicité d'ontologies.
- Hétérogénéité des modèles de représentation : les ontologies sont représentées par des formalismes différents (logiques de descriptions, réseaux sémantiques...). Chaque formalisme a ces propres règles et artéfacts pour représenter les concepts et les relations d'un domaine.
- Multiplicité de langages : il existe une multitude de langages de description d'ontologies (KIF, RDF, SHOE, OWL..etc). L'intéropérabilité entre ces langages est non assurée et représente une source considérable d'hétérogénéité.
- Modélisation : constitue elle aussi une source d'hétérogénéité. En effet, la modélisation d'ontologies ne se base pas sur des règles de l'art claires et précises. Ceci pose des problèmes à caractère conceptuel ; l'ontologie reflète la vision du concepteur sur un monde d'intérêt. Ainsi, les concepts, les relations, les contraintes et le niveau de granularité est tributaire à cette vision. Il n'est pas étonnant donc d'avoir de modèles différents pour une même ontologie couvrant la même réalité.

- Multi-linguisme : Les concepts et les relations sont décrits dans le vocabulaire d'un langage naturel, ceci fait surgir des problématiques inhérentes au traitement du langage naturel comme les synonymes, acronymes. Le choix d'un langage naturel unique, comme l'Anglais, sort du cadre technique ; la question a une dimension plutôt éthique.

1.2 Contributions

A l'issue des réflexions menées conjointement avec l'équipe franco-québécoise, nous avons proposé une approche d'alignement d'ontologies OWL-Lite : l'approche s'articule sur deux axes principaux : le premier consiste à représenter une ontologie OWL-Lite sous forme d'un graphe dit OL-Graph et le deuxième s'intéresse à la mise en œuvre d'un modèle de calcul de similarité entre les entités d'ontologies OWL-Lite par catégorie.
Le présent travail a pour objectif de développer les propositions à un niveau suffisant de détail pour compléter et valider les deux propositions et enfin élaborer un environnement d'alignement et de visualisation d'ontologies OWL-Lite.

Sur le plan théorique nous avons complété et affiné le méta-modèle de représentation d'ontologies OWL-Lite nommé OL-Graph. L'élaboration de ce modèle constitue le socle du modèle de calcul de similarité. En effet, l'OL-Graph représente une traduction de la grammaire OWL-Lite à une représentation graphique qui reflète par ces artéfacts les entités et les relations d'une ontologie OWL-Lite. Ainsi, pour pouvoir représenter toute la grammaire d'OWL-Lite, il faudrait d'une part penser à l'interprétation sémantique des différents constructeurs du langage OWL-Lite et d'autre part trouver la représentation adéquate par les artefacts d'OL-Graph.

L'approche d'alignement proposée s'inspire de différents travaux sur la structuration de bases de connaissances [Valtchev et Euzenat, 1997] et l'alignement d'ontologies [Euzenat et Valtchev, 2004] exprimées sous OWL. Le calcul de la mesure de similarité se base sur les techniques d'analyse terminologique, contextuelle et structurelle des entités pour définir une mesure de similitude entre les entités par catégorie. Nous étions amené à apporter des ajustements sur le modèle de calcul de similarité original. En effet, l'application du modèle pénalise le calcul pour les couples d'entités dont l'une est peu décrite par rapport à l'autre. Ainsi, il faudrait uniformiser le calcul de similarité pour tenir en considération un tel cas.

L'implémentation de notre approche d'alignement a donné naissance au système OLA (Ontology Lite Alignment). Le système représente le noyau d'un environnement ouvert et intégré d'alignement et

de visualisation d'ontologies décrites en language OWL.

Dans un soucis de partager nos idées et nos travaux, nous avons publié *OLA* sous le source Forge dans sa première version Beta et participé à la compétition sur l'alignement d'ontologies [EON, 2004].

1.3 Plan du mémoire

Dans un premier chapitre nous définissons le cadre de l'étude et cernons la problématique ; nous présentons au lecteur les notions et le vocabulaire sur le Web sémantique, le problème d'alignement et le langage OWL. Ensuite, dans le deuxième chapitre, nous proposons une étude de l'art sur l'alignement en général. Nous avons recensé les démarches d'alignement d'ontologies les plus citées dans la littérature dont pour chacune ses principes de base. Dans le troisième chapitre, nous décrivons notre approche d'alignement, les techniques sous-jacentes ainsi que les améliorations que nous y avons apporté. Dans le quatrième chapitre, nous abordons les questions inhérentes à la mise en œuvre de notre approche d'alignement et les expérimentations conduites. Finalement, nous concluons sur ce mémoire par des recommandations sur les possibilités d'amélioration et l'extension de l'approche, nous proposons de même des suggestions sur les pistes de recherche éventuelles.

Chapitre 2

Mise en contexte

Dans ce chapitre, nous situons le lecteur dans le contexte général de notre travail. Ainsi, nous introduisons les concepts autour du Web sémantique en donnant quelques définitions des mots clés et des concepts théoriques. Nous allons sensibiliser le lecteur à l'importance de l'alignement par des cas d'application dans des domaines cruciaux. En outre, et pour aider à comprendre les fondements du langage *OWL-Lite*, nous donnons une brève introduction sur les logiques de descriptions ainsi que les éléments grammaticaux du langage OWL-Lite.
A la fin de ce chapitre, le lecteur constituera une idée sur le cadre général du sujet et sur la problématique en question.

2.1 Le Web sémantique

Le Web actuel décrit son contenu à travers des textes formulés dans des langages naturelles lisibles que par des humains. Une forme statique et très peu adaptée quand il s'agit de traiter des ressources sur Web par des non-humains (les machines, les agents, les application). A l'encontre du Web "syntaxique", le Web sémantique ajoute une couche sémantique au Web. Ainsi, les ressources sont décrites sous forme d'ontologies : une façon de modéliser et de décrire les ressources - décrire les concepts et leurs relations sous un langage de description d'ontologies utilisant un vocabulaire commun - devenue de plus en plus adoptée. Ainsi, le Web sémantique ouvre la voie vers la construction d'un Web doté des outils ayant la possibilité de raisonner et de collaborer. Les horizons d'application qui s'annoncent sont nombreux et très prometteurs.
Les recherches dans le domaine du Web sémantique ont marqué un progrès considérable. Ainsi, l'apparition des standards comme XML, RDF, et le langage de description d'ontologies OWL ont permis la description des ressources sur le Web et l'expression des relations entre les concepts. Cependant, des problématiques majeures heurtent au développement du Web sémantique, plus particulièrement la hétérogénéité des ontologies qu'est au centre des défis qu'il faudrait surmonter.

2.2 Les ontologies sur le Web

Les ontologies constituent le socle du Web sémantique. Grâce aux ontologies, les ressources ne sont plus décrites par des textes statiques mais plutôt, mobilisées et exprimées sous une forme permettant de refléter les concepts et leurs relations dans un vocabulaire commun facilitant ainsi le partage de la connaissance et son traitement par des non-humains.

2.2.1 Définition d'une ontologie

Le concept *ontologie* trouve son origine dans la métaphysique [Chandrasekaran et al., 1999] où le terme fait référence au sujet de l'existence. Le mot a été utilisé ensuite par la communauté de l'intelligence artificielle. Ainsi, plusieurs définitions du concept ont été proposées. Les plus souvent citées dans littérature sont :

1- An ontology[1] [Swartout et al., 1996] is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base.

[1] L'ontologie est un ensemble de concepts hiérarchiquement structurés qui décrivent un domaine d'intérêt qui peut être utilisé comme une squelette de base pour une base de connaissances.

2- An ontology[2][Noy et McGuinness, 2001] defines a common vocabulary for researchers who need to share information about a domain. It includes machine interpretable definitions of the basic concepts and the relations among them.

3- Une ontologie définit les termes utilisés pour décrire et représenter un domaine de connaissance. Les ontologies sont utilisées pour partager un domaine de connaissance, ou un domaine d'intérêt spécifique, comme la médecine, la manufacture d'objets, l'immobilier etc.. Les ontologies comprennent des définitions, traitables par machine, de concepts de bases du domaine et leurs relations. Elles encodent les connaissances afférentes à un domaine mais qui peuvent aussi s'étendre à d'autres domaines. De cette façon, elles doivent rendre ce savoir ré-exploitable. Le mot ontologie a été utilisé pour décrire des artéfacts de différents degrés de complexité. Ceci va de simples taxonomies, comme la hiérarchie yahoo, aux schémas de méta-données comme le Dublin Core, et les théories logiques [W3C].

La définition [Gruber, 1995], nous semble, caractériser le mieux l'essence d'une ontologie :

4- An ontology[3] is a formal, explicit specification of a shared conceptualisation.

2.2.2 Exemple d'une ontologie réelle

La figure FIG.2.1 montre l'exemple d'une partie d'une ontologie réelle, tirée de l'ontologie[4] de l'université de stanford qui décrit une bibliographie. Dans cet exemple, nous avons évité de représenter l'ontologie dans un modèle ou dans un langage de représentation bien particulier. Ainsi, nous avons adopté une représentation simple qui met en évidence principalement les éléments du domaine sous forme d'une taxonomie hiarchisée de concepts.

2.2.3 Quelques règles de conception d'ontologies

En l'absence des règles de l'art standards et claires dans le processus de modélisation, de construction et de maintenance des ontologies, les critères que doit vérifier une bonne ontologie restent un sujet contro-

[2]L'ontologie définie un vocabulaire commun pour un partage de l'information d'un domaine d'intérêt. Elle inclut les définitions de concepts élémentaires et les relations entre eux sous une forme lisible par les machines.

[3]Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée. Le mot *conceptualisation* signifie un modèle abstrait d'un domaine d'intérêt dont les concepts pertinents ont été identifiés et recensés. Le mot *explicite* implique que les types de concepts ainsi que les contraintes exprimés sur ces concepts soient explicitement définies. Le mot *formelle* indique que l'ontologie doit être décrite dans un format (ou un langage) lisible par les machines. Le mot *partagée* fait référence au fait que l'ontologie doit capturer une partie consensuelle de la connaissance acceptée par toute ou une large partie de la communauté derrière l'ontologie.

[4]http ://www.ksl.stanford.edu/knowledge-sharing/ontologies/html/bibliographic-data/)

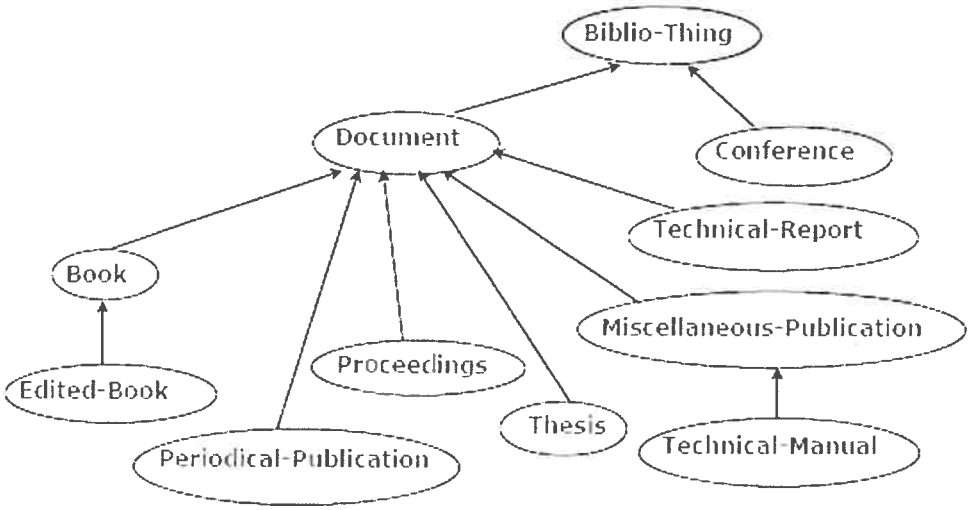

FIG. 2.1 – Exemple d'une partie d'une ontologie réelle : université de Stanford

versé. Cependant, l'ingénierie des ontologies met l'accent sur les qualités suivantes [Gruber, 1995] :

- **Clarté** : l'ontologie doit communiquer effectivement le sens des termes et des axiomes définis. Ainsi, les définitions doivent être objectives, claires et indépendantes du contexte social et des considérations techniques.
- **Cohérence** : les définitions des concepts et des axiomes doivent être logiquement consistantes, cohérentes de telle sorte qu'elles sanctionnent les inférences sans contredire les autres définitions des autres concepts.
- **Extensibilité** : avoir la possibilité de définir de nouveaux termes en utilisant le vocabulaire partagé et réutilisable déjà existant. Ceci sans être obligé de revoir les définitions existantes.
- **Biais d'encodage minimal** : la conceptualisation doit être spécifiée au niveau de- connaissance (knowledge level) sans dépendre d'un encodage d'implémentation particulier. Ce biais d'encodage doit être minimal pour permettre le partage, et la réutilisation de la connaissance le plus efficacement et largement possible.
- **Engagement ontologique minimal** : l'ontologie doit satisfaire l'engagement ontologique minimal "minimal ontological commitment" pour supporter la réutilisation et le partage de la connaissance entre les tiers c'est-à-dire utiliser le plus restreint nombre possible de termes et le minimum d'assomptions sur le monde modélisé. Ceci permet la communication de la connaissance sur le monde modélisé, tout en permettant aux tiers la liberté de spécialiser ou d'instancier l'ontologie

au besoin.

2.2.4 Les ontologies sur le Web

Il faut signaler que le développement d'ontologies autour du Web a un caractère pragmatique. En effet, l'utilisation des ontologies représente une solution au problème de la description et la représentation des ressources sur le Web. Ainsi, le gain de leur utilisation va au delà d'une simple organisation de la connaissance à l'échelle du Web : il touche à des aspects plus importants liés au traitement automatique des connaissances, le partage et l'exploitation intelligente des ressources. En effet, les ontologies définissent un vocabulaire commun sous une forme lisible permettant de faire du raisonnement. Par conséquent, elles sont de plus en plus adoptées dans les domaines de la gestion de l'information, le e-commerce, la fouille d'informations sur le Web et bien d'autres.

2.3 Besoin d'alignement

Face au problème d'hétérogénéité d'ontologies, il devient impératif de penser à des moyens pour assurer leur intéropérabilité pour pouvoir les faire collaborer et permettre aux composants intelligents d'opérer à l'échelle du Web. Il s'agit en d'autres termes de trouver des procédés pour réconcilier entre les ontologies.
A titre illustratif, l'ontologie[5] FIG. 2.2 décrit à l'instar de l'ontologie présentée dans l'exemple précédent, le même domaine d'intérêt celui d'une bibliographie selon WordNet. Cependant, elle ne manifeste pas la même classification de concepts, bien que des concepts similaires -ayant la même signification- puissent être facilement dégagées. Ainsi, les résultats des requêtes de recherche renvoyés par des engins de recherche automatiques sur le contenu des deux ontologies, pourraient être plus pertinantes si les deux ontologies sont fusionées et vues comme une seule. Ceci représente, en fait, un des aspects où l'alignement est nécessaire pour pouvoir déterminer les concepts similaires entre les deux ontologies. Comme on verra ultérieurement, le recours à l'alignement touche divers domaines d'application.

A priori, il existe deux techniques de réconciliation entre les ontologies.

- La ***fusion*** : elle consiste à intégrer deux ontologies pour produire une nouvelle regroupant les concepts, les relations et les instances des ontologies originales. Les ontologies fusionnées perdent en revanche leurs existences.
- L' ***alignement*** : la technique s'intéresse plutôt à trouver les entités similaires des ontologies en question.

[5]L'ontologie complète est sous le lien http ://www.cogsci.princeton.edu/ wn/w3wn.html

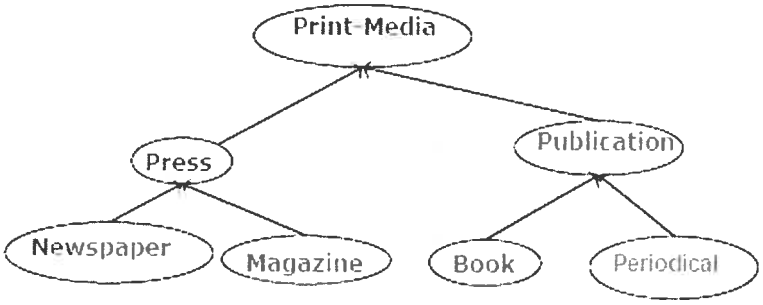

FIG. 2.2 – Exemple d'une partie d'une ontologie réelle de la bibiographie de WordNet.

2.3.1 Définition d'alignement

L'alignement d'ontologies consiste à chercher les concordances entre les concepts, les relations et les individus des diverses ontologies. Le but c'est de trouver les points de jonctions (les entités en commun.) qui permettront de concevoir des ponts entre ces ontologies, ou de procéder à d'autres opérations de manipulation comme la fusion ou l'intégration partielle. Pour cette raison, l'alignement est considéré comme une technique à base de toute autre opération de réconciliation entre les ontologies.

Cependant, trouver les concepts en commun manuellement est une tache fastidieuse, pratiquement infaisable dans le cas des ontologies volumineuses. Par conséquent, trouver des moyens pour automatiser ce processus aura un apport important dans le domaine d'ingénierie d'ontologies plus particulièrement celui du Web sémantique.

2.3.2 Quelques applications potentielles

L'expansion du Web sémantique et l'adoption des ontologies comme moyen de décrire les ressources sur le Web a accentué davantage ce besoin. Ainsi, les domaines d'applications manifestant le besoin aux outils d'alignement sont multiples comme l'E-commerce, la fouille d'information. Le but par ces cas d'application est de montrer l'intérêt de l'alignement en général -celui des ontologies en particulier-.

2.3.2.1 Cas 1 : Communicaiton entre les agents

Les agents mobiles [Wiesman et al., 2002] sont des entités caractérisées par leur mobilité, leur autonomie et leur capacité d'intéraction. Les agents mobiles connaissent une utilisation accrue dans les

domaines du E-commerce, la fouille de l'information etc.
Suivant le modèle courant, dans le cas des agents mobiles dits cognitifs, le comportement de ces agents est considéré comme un ensemble de croyances, désirs et attentions exprimés symboliquement dans des langage inspérés de "speech-act" pour intéragir entre eux. Ces langages déterminent "l'enveloppe" des messages interchangés entre les agents et leur permettent de situer ces messages dans un contexte d'intéraction. Cependant, ces langages ne specifient pas le contenu des messages qui peut être exprimé dans des langages de représentation de connaissances différents. Par conséquent, pour que les agents se comprennent mutuellement, il faudrait qu'ils soient en mesure de traduire leurs messages et de les aligner via l'intégration des ponts d'axiomes dans leur propres modèles [Euzenat et al., 2004a].
Une première solution à ce problème consiste à adopter un protocole d'alignement d'ontologies, qui se déclenche à la réception d'un message d'un autre agent faisant référence à une ontologie hétérogène.

2.3.2.2 Cas 2 : les services Web sémantiques

Les services Web [Magini et al., 2003] sont des processus qui exposent leurs interfaces pour offrir des fonctionalités à d'autres applications pour des raisons de complémentarité. Les services Web sémantiques offrent un moyen plus riche et plus précis pour décrire les services via l'utilisation des langages de représentation de connaissances et d'ontologies. La découverte et l'intégration des services Web consiste à découvrir un service Web particulier et/ou combiner plusieurs services pour fournir un autre service plus complexe. Toutefois, la description de tels services fait référence à des ontologies multiples généralement exprimées dans des modèles et des langages hétérogènes. Par conséquent, l'opération d'intégration des services nécessite l'établissement de correspondances entre les termes d'ontologies.

A tire d'illustration, pennons le cas de deux services Web A et B dans le contexte du E-commerce. En effet, les deux services pouvent ne pas partager le même modèle (ontologie) pour décrire les fonctionalités qu'ils supportent. Par exemple, le service B fourni la possibilté d'annuler une commande alors que le A ne la supporte pas, ou le cas inverse, le service A implémente la fonctionalité de changer une commande alors que B ne la fourni pas. Dans ce genre de situation, pour que les deux services Web s'engagent eventuellement dans des transactions de vente-achat, une vérification des fonctionalités suportées exige une reconciliation entre les deux ontologies à l'avance pour déterminer les fonctionalités en commun. Ceci revient à aligner les deux ontologies, en temps réel à l'aide d'outils d'alignement.

2.3.2.3 Cas 3 : les réseaux P2P

La technologie des réseaux Peer-to-Peer [Li et Clifton, 1994] a été déployée pour le partage des resources entre les applications (exemple KaZaA, Napster). Ces réseaux se caractérisent par leur dynamisme et leur flexibilité. Toutefois, les nœuds des réseaux sont autonomes en contenu, en langages de

description des ressources et en la manière de les échanger. En effet, ils peuvent adopter des terminologies disparates ou faire usage à des ontologies. Pour assurer un échange efficace d'information entre les nœuds, il faudrait identifier et caractériser les relations de leurs schémas/ontologies d'une manière instantanée lors de l'échange de données.

2.3.2.4 Cas 4 : la recherche dans les catalogues d'E-commerce

Les applications B2B (le cas d'*Amazon*, *e-Bay* etc) ont recours à des catalogues électroniques sous forme de taxonomies hiérarchisées de concepts avec leurs attributs afin de stocker et d'indexer le volume important de produits. En effet, des standards comme *UNSPSC*[6] est largement utilisé pour représenter les besions du vendeur, alors que le standard *ecl@ss*[7] est utilisé pour représenter les besions de l'acheteur. D'un autre côté, il faudrait permettre à des applications de fouiller et de restituer de l'information pertinente à partir de ces catalogues dont le contenu n'est pas nécessairement décrit dans le même modèle (le cas des deux standards UNSPESC et ecl@ss). Pour ce faire, il faudrait que ces catalogues soient alignés [Cui et al., 2003] moyenant des procédés automatiques pour que les outils et les engins de recherche puissent accéder à leurs contenus.

2.4 Introduction aux logiques de description

L'importance des langages de représentation de connaissances dans le domaine des ontologies sur Web n'a guère besoin d'être rappelée. Un Web sémantique réussi, doté de mécanismes intelligents dépend d'ontologies bien modélisées, bien structurées et décrites dans des langages ayant les mécanismes d'inférence permettant des exploitations et des vérifications automatiques.

Vu leurs caractéristiques ainsi que les avantages qu'elles manifestent par rapport à d'autres formalismes de représentation de connaissances [Baader et al., 2003], les logiques de description ont été adoptées pour fournir les fondements théoriques du langage de description d'ontologies OWL (Ontology Web Language).

2.4.1 Aperçu sur les logiques de descriptions

Les recherches dans le domaine de la représentation de connaissances s'intéressent particulièrement aux moyens de décrire les objets du monde réel dans un niveau d'abstraction qui permettra de construire des systèmes intelligents. Les logiques de description[Baader et Nutt, 2002] sont les descendants de ce

[6]http ://www.unspsc.org/

[7]http ://www.eclass-online.com/

qu'on appelle "les réseaux d'héritage structurés". L'idée clé des LD est d'utiliser des concepts et des rôles atomiques (ou prémitives) et un nombre de constructeurs pour former des concepts et des rôles plus complexes. Un concept est une relation unaire qui peut correspondre à une classe ou à un individu alors qu'un rôle représente un prédicat binaire qui relie deux concepts atomiques ou composés.

Les logiques de descriptions ont connu des extensions d'expressivité c'est-à-dire l'ensemble des constructeurs utilisés. Ceci a fait des LD une famille de langages dont chacun utilise un ensemble de constructeurs bien déterminé comme *ALC*. Cette famille représente la forme la plus basique, où les concepts sont définis à partir d'autres concepts atomiques utilisant les relations binaires de premier ordre comme la conjonction, la négation etc.

2.4.2 Éléments syntaxiques et sémantiques des LD

Dans une base de connaissances formalisée sous les logiques de descriptions, on peut facilement distinguer entre deux catégories de connaissances [Baader et Nutt, 2002] :

- Les *TBox* contiennent les connaissances *intentionnelles* sous forme d'une terminologie (d'ou le terme TBox) construite via les déclarations qui décrivent en général les propriétés des concepts.
- Les *ABox* contiennent les connaissances *extensionnelles* désignées aussi par le terme connaissances *assertionnelles*. Elles expriment les connaissances liées aux individus du domaine d'intérêt.

Les connaissances intentionnelles sont pensées à ce qu'elles ne changent pas au fil du temps. Cependant, les connaissances assertionnelles sont liées à des cas particuliers et donc susceptibles à des changements.

2.4.3 Raisonnement dans les LD

Le but de la représentation de la connaissance est de fournir une formalisation structurée des connaissances ainsi que de permettre de raisonner sur ces connaissances : avoir des mécanismes d'extraction des connaissances implicitement exprimées d'une part et permettre la vérification de la cohérence des connaissances et la validité des assertions déjà exprimées d'une autre part. Les techniques de raisonnement dans les LD ont connu des développements importants dans le temps, commençant par des techniques ad hoc les plus spécialisées jusqu'aux méthodes totalement généralisées. Ainsi, on identifie trois catégories de techniques :

- **Les techniques structurées** : ces techniques servent à raisonner sur des concepts dans le cas des LD moins expressives ou ce qu'on appelle LD faibles.
- **Les techniques à base de tableaux** : elles ont été proposées comme des techniques plus générales pour raisonner sur des LDs plus expressives où la notion du système contraint a été introduite.
- **Les techniques à base d'automates** : elles sont considérées les plus réussies pour raisonner sur les LDs qui incluent les formes fixes.

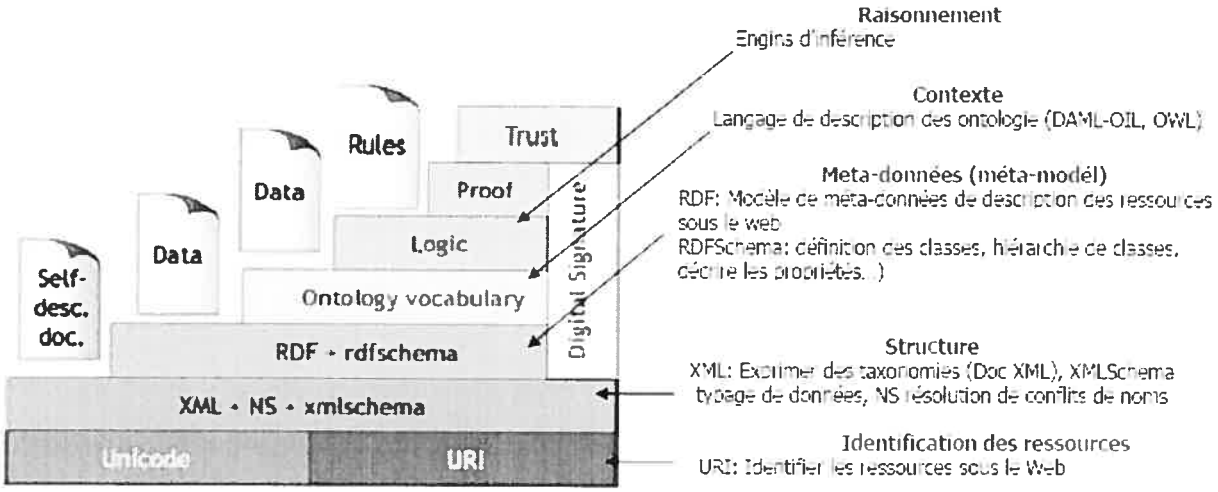

FIG. 2.3 – L'architecture du Web sémantique : selon Tim beneer Lee

2.5 Langage de description d'ontologies sur le Web (OWL)

Le langage de description d'ontologies OWL [Smithv et al., 2004] est le successeur d'un ensemble de technologies qui ont commencé avec le langage XML, passant par RDF, RDFSchema et arrivant au langage (Ontology Web Language) OWL. La figure FIG. 2.3 représente l'architecture du Web sémantique. Elle résume entre autre, les différents langages et schémas de description de la connaissance sous le Web.

2.5.1 l'URI

L'URI est l'acronyme de l'expression "Unified Resource Identifier". C'est une chaîne de caractères permettant l'identification des ressources d'une manière standard et unique sur le Web. L'URI représente aussi le mot générique pour désigner la façon d'adressage des ressources sur le Web. Il englobe l'URL (Unified Resource Locator), URN (Unified Resource Name), URC (Unified Resource Classification).

2.5.2 XML et XMLSchema

L'émergence d'XML, développé par le W3C comme un standard, constitue une première étape vers un langage de description d'ontologies sur le Web. XML est destiné plus particulièrement à l'échange de données (des documents) sur le Web, rend possible la description des concepts et leurs attributs ainsi que leur hiérarchisation sous forme de taxonomies à l'aide des balises. Cependant, l'XML reste très léger

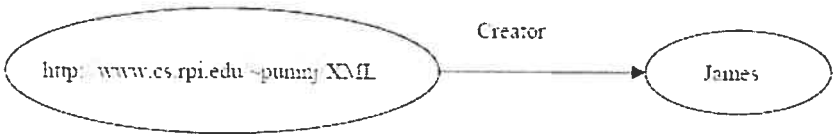

FIG. 2.4 – Description d'une ressource selon le modèle RDF

et limité quand il s'agit d'exprimer des contraintes et des relations entre des concepts ou des axiomes. Ces lacunes ont motivé le développement d'autres langages comme DTD (Document Type Definition) et XMLSchema. En effet, DTD et XMLSchema offrent des grammaires qui permettent de spécifier des restrictions sur la structure des documents (taxonomies) et préciser les noms et le typage des attributs. Cependant, ils n'offrent aucun mécanisme pour exprimer des contraintes ou des relations sémantiques entre les concepts.

2.5.3 RDF et RDFSchema

Le RDF (Ressources Description FrameWork) est un formalisme standard développé par le W3C pour décrire les ressources sur le Web [Decker et al., 2000]. Le RDF ne représente pas un langage mais un méta-modèle pour décrire les ressources sur le Web en utilisant la syntaxe d'XML. Il est équivalent au formalisme des réseaux sémantiques. Ainsi, et suivant ce modèle, chaque ressource sur le Web peut être décrite sous forme d'un triplet composé d'une ressource, d'une propriété et d'une valeur (voir l'exemple donné par la figure FIG.2.4).

1. **ressource** : décrite par des expressions de RDF et nommée souvent par des URIs. Elle peut être une page Web ou une partie d'une page définie par une balise XML (la page Web identifiée par l'URI
   http ://www.cs.rpi.edu/ puninj/XMLJ/ dans la FIG.2.4).
2. **propriété** : représente un aspect particulier, une caractéristique, un attribut ou une relation qui décrit la ressource (la propriété *Creator* dans la figure FIG.2.4).
3. **valeur** : peut être à son tour une ressource, assignant une valeur à une propriété d'une ressource spécifique (la valeur *James* dans la FIG.2.4).

A priori, le modèle de données RDF ne permet ni la définition de relations entre les propriétés (les attributs et les ressources) ni l'expression d'assomptions sur un domaine bien particulier. Le RDFSchema

offre alors un ensemble de constructeurs pour définir des classes, des propriétés ainsi que d'exprimer des relations de spécialisation entre elles.

2.5.4 Langage DAML+OIL

DAML+OIL [Connolly et al., 2001] est le fruit d'un projet commun de la fusion de deux langages : l'OIL (Ontology Interchange Language) développé par l'union européenne et DAML (DARPA Agent Modeling Language) développé par les États Unis dans le cadre du projet DARPA (The Defense Advanced Research Projects Agency). DAML+OIL est construit sur le modèle RDFs et basé sur le modèle théorique des logiques de descriptions. Il est considéré le premier langage à fournir des mécanismes d'inférence sur les concepts d'ontologies.

2.5.5 Langage OWL

Le langage OWL est le descendant de l'ensemble des langages ci-haut mentionés, développé par le W3C comme une révision du langage DAML+OIL. OWL [Smithv et al., 2004] permet de décrire les structures d'un domaine en terme de classes et de propriétés dans une approche similaire à celle de l'approche orienté objet. L'objectif visé par l'OWL est de permettre de décrire les ontologies dans un langage commun permettant ainsi le partage, l'import et l'export d'ontologies ainsi que de pallier au problème d'hétérogénéité des autres langages.

A l'instar de DAML+OIL, OWL s'appuie sur les fondements théoriques des logiques de descriptions : les classes et les propriétés dans OWL sont équivalentes aux concepts et aux rôles respectivement. OWL fournit une multitude d'opérateurs pour définir des concepts (classes, objets...) plus complexes à partir des concepts élémentaires ou combinés. Cet aspect s'ajoutant à la possibilité de supporter toute sorte d'axiomes, dote le langage d'une grande expressivité. OWL offre aussi des mécanismes d'inférence et de raisonnement sur les ontologies à savoir la vérification de la consistance de classes, de sous-assomptions et d'autres types d'assertions pour supporter l'intégration et le déploiement d'ontologies.
OWL vient en trois versions dont le degré d'expressivité et de décidabilité sont mutuellement inverses. Le but est de permettre un choix de langage offrant un compromis entre le degré d'expressivité et la qualité d'inférence permise. Les trois versions du langage sont :

- **OWL-Lite** : il est destiné aux cas de modélisation se limitant à une classification simple de concepts et de contraintes. Il supporte un sous ensemble de constructeurs comparativement à OWL-DL et OWL-Full.
- **OWL-DL** : permet une plus grande expressivité faisant usage un l'ensemble de constructeurs du

langage OWL. En revanche la complétude et la décidabilité des algorithmes d'inférences ne sont pas garanties.

- **OWL-Full** : il est adressé à des utilisateurs qui cherchent un maximum d'expressivité et la liberté syntaxique de RDF. Ainsi, une classe par exemple, peut être considérée comme une collection d'individus ou comme un individu à part entière. OWL-Full permet par conséquent, d'augmenter le sens du vocabulaire déjà défini de RDF et celui d'OWL. Cependant, pareil au OWL-DL, la décidabilité et la complétude des inférences ne sont pas assurées.

2.5.6 Les éléments syntaxiques d'OWL-Lite

Les éléments syntaxiques d'OWL-Lite constituent la grammaire du langage. Dans cette section on présente les éléments élémentaires du langage en donnant pour chacun un exemple illustratif.

2.5.6.1 Classe

La classe peut être explicite ou anonyme. Elle regroupe un ensemble d'individus appartenant au même concept.
La classe *Thing* représente la classe la plus générique dans le langage OWL alors que la classe *Nothing* est la plus spécifique. Les classes OWL sont décrites par des descriptions qui peuvent être combinées avec des axiomes de classes.

Exemple d'une classe simple dans OWL

```
<owl:Class rdf:ID="University">
</owl:Class>
```

2.5.6.2 Descriptions de la classe

- **Intersection** : Une classe peut être définie comme étant l'intersection de deux ou plusieurs classes ou descriptions à l'aide du constructeur *owl :intersecitonOf.*

Exemple d'une intersection de deux classes

```
<owl:Class rdf:ID="Woman">
   <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Female"/>
      <owl:Class rdf:about="#Human"/>
   </owl:intersectionOf>
```

```
</owl:Class>
```

La classe *"Woman"* est l'intersection de deux classes *Female* et *Human*.

- **Restriction** : La restriction est un cas spécial d'une description. Elle décrit une classe anonyme dont les individus satisfont la restriction. On distingue deux types de restrictions ; les *restrictions par valeur* et *les restrictions par cardinalités*. Une restriction par valeur exprime une contrainte sur le *range* d'une propriété alors qu'une restriction par cardinalités exprime la contrainte sur le nombre de valeurs qu'une propriété peut avoir dans le contexte de cette description.

Exemple d'une restriction dans OWL

```
<owl:Restriction>
    <owl:onProperty rdf:resource="(some property)"
</owl:Restriction>
```

Exemple d'une restriction par valeur

```
<owl:class>
   <owl:Restriction>
       <owl:onProperty rdf:resource="#hasParent"/>
       <owl:allValuesFrom rdf:resource="#Human"/>
   </owl:Restriction>
</owl:class>
```

L'exemple décrit une classe anonyme dont les individus sont pour lesquels la propriété "hasParent" a toutes les valeurs de la classe *Human*.

Exemple d'une restriction par cardinalités

```
<owl:Restriction>
    <owl:onProperty rdf:resource="#hasParent"/>
    <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2
    </owl:maxCardinality>
</owl:Restriction>
```

L'exemple décrit une classe anonyme dont tous les individus sont ceux qui ont la propriété "hasParent" liée à au moins une valeur de la classe *Human*.

2.5.6.3 Axiomes sur les classes

- **Relation d'équivalence** : Dans OWL on peut définir une relation d'équivalence entre les classes. La relation d'équivalence signifie que les deux classes ont les mêmes instances. Elle sert aussi à

créer des classes synonymes.

Exemple d'une relation d'équivalence entre deux classes

```
<owl:Class rdf:ID="baby">
  <owl:equivalentClass>
      <owl:Class rdf:ID="child">
      </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

- **Sous-classement** : une sous classe représente une spécialisation d'une classe donnée. La déclaration d'une sous classe se fait par le constructeur *owl :subClassOf*.

Exemple d'une sous classe

```
<owl:Class rdf:ID="Food">
     <rdfs:subClassOf>
        <owl:Class rdf:ID="SpicyRedMeat" />
     </rdfs:subClassOf>
</owl:Class>
```

2.5.6.4 Propriétés

Les propriétés dans le langage OWL permettent d'exprimer des faits généraux sur les instances de classes et des faits spécifiques sur les individus. Les propriétés sont de deux types :

- **Propriété de nature "objet"** : elle représente une relation entre les instances de deux classes.

Exemple d'une propriété de nature objet

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
    <rdfs:domain rdf:resource="#Wine"/>
    <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>
```

Dans l'exemple la propriété *madeFromGrape* est définit par rapport à la classe *Wine* (domain) dont les valeurs de ces instances sont liées aux instances de la classe *WineGrape* (range).

- **Propriété de nature "type de donnée"** : elle représente une caractéristique simple dont les valeurs n'ont pas d'identité entre les instances d'une classe et un littéral de RDF ou un type de

donnée XML.

Exemple d'une propriété de nature "type de donnée"

```
<owl:DatatypeProperty rdf:ID="hasColor">
  <rdfs:domain rdf:resource="#SpicyRedMeat" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
</owl:DatatypeProperty>
```

La propriété *hasColor* est définit par rapport à la classe *SpicyRedMeat* (domain) dont les valeurs d'instances sont de type chaîne de caractères (range).

2.5.6.5 Sous propriété

Une sous propriété représente une spécialisation de la propriété générale ayant le même type. Elle est déclarée par le constructeur *owl :subProperty*.

Exemple d'une sous propriété

```
<owl:ObjectProperty rdf:ID="body">
   <rdfs:subPropertyOf rdf:resource="#part" />
   <rdfs:domain rdf:resource="#Camera" />
   <rdfs:range rdf:resource="#Body" />
</owl:ObjectProperty>
```

2.5.6.6 Axiomes sur les propriétés

- **Relation d'équivalence** : elle est exprimée par le constructeur *owl :EquivalentProperty*

  Exemple d'une relation d'équivalence entre deux propriétés

```
<owl:ObjectProperty rdf:ID="body">
   <rdfs:equivalentProperty rdf:resource="#part" />
   <rdfs:domain rdf:resource="#Camera" />
   <rdfs:range rdf:resource="#Body" />
</owl:ObjectProperty>
```

- **Relation inverse** : une relation définit normalement une relation d'un seul sens du domaine au range, la relation inverse permet d'exprimer une propriété comme l'inverse d'une autre. Elle est

introduite par le constructeur *owl :inverseOf*

Exemple d'une relation inverse entre deux propriétés

```
<owl:ObjectProperty rdf:ID="hasChild">
    <owl:inverseOf rdf:resource="#hasParent"/>
</owl:ObjectProperty>
```

- **Relation symétrique** : une relation de symétrie vérifie que pour tout couple (x, y) ayant la propriété P, le couple (y,x) est aussi une instance de la propriété P. La relation de symétrie entre deux propriétés est exprimée par le constructeur *owl :SymmetricProperty*

Exemple d'une relation symétrique entre deux propriétés

```
<owl:SymmetricProperty rdf:ID="friendOf">
   <rdfs:domain rdf:resource="#Human"/>
   <rdfs:range  rdf:resource="#Human"/>
</owl:SymmetricProperty>
```

2.5.6.7 Individu

L'individu représente une instance d'une classe. Il peut être déclaré par plusieurs syntaxes.

Exemple d'un individu

```
<SpicyRedMeat rdf:ID="BeefCurry" />
```

L'exemple est une définition d'un individu nommé *BeefCurry* qu'est une instance de la classe *SpicyRedMeat*.

2.5.6.8 Relations entre individus

Les relations entre les individus sont de deux types :

- **Relation d'identité** : elle est exprimée par le constructeur *OWL :sameAs*. Ceci signifie que deux individus sont identiques.

Exemple d'une relation d'identité entre deux individus

```
<rdf:Description rdf:about="#football_game">
   <owl:sameAs rdf:resource="#Soccer"/>
</rdf:Description>
```

- **Relation de différence** : elle permet d'exprimer que deux individus sont mutuellement différents. La relation est exprimée par le constructeur *owl :differentFrom*. Le constructeur *owlAllDifferent-From* dans le contexte de plusieurs individus

Exemple d'une relation de différence entre plusieurs individus

```
 <owl:AllDifferent>
   <owl:distinctMembers rdf:parseType="Collection">
       <Opera rdf:about="#Don_Giovanni"/>
       <Opera rdf:about="#Nozze_di_Figaro"/>
       <Opera rdf:about="#Cosi_fan_tutte"/>
       <Opera rdf:about="#Tosca"/>
       <Opera rdf:about="#Turandot"/>
       <Opera rdf:about="#Salome"/>
       </owl:distinctMembers>
</owl:AllDifferent>
```

2.6 Conclusion

Dans ce chapitre, nous avons passé en revue les définitions de notions jugées essentielles pour cerner le champ de l'étude et comprendre les principes à base de méthodes d'alignement. Ainsi, nous avons défini le Web sémantique comme une extension du web "syntaxique" et la problématique qui freine son expansion. De même, nous avons présenté les ontologies et la problématique d'hétérogénéité. Vu que notre approche s'intéresse à l'alignement d'ontologies décrites sous le langage OWL-Lite, une présentation des logiques de descriptions est justifiée, du moment qu'elles sont à base du langage OWL-Lite.

Le choix de ce dernier par rapport à OWL-DL et Full se justifie d'un côté par de sa simplicité. En effet, il constitue un bon point de départ pour mettre en œuvre et valider notre approche d'alignement, d'où nous pourrons par la suite, la généraliser sur le DL et le F ull. D'un autre côté, la plus grande disponibilité des ontologies valides décrites sous OWL-Lite constituait un élément nécessaire pour effectuer des tests et des études comparatives.

Le chapitre suivant met l'accent sur les travaux effectués sur l'alignement d'ontologies et les techniques adoptées pour élaborer des outils automatiques d'alignement. La première section du chapitre fait le survol de différentes méthodes de rapprochement du calcul de la similarité entre les couples d'entités. La deuxième section est consacrée à des cas de systèmes d'alignement d'ontologies à base des techniques présentées. Ainsi, des observations[8] ont été rapportées pour certaines approches .

[8]Certaines de ses observations sont celles citées par les auteurs des articles de ces systèmes.

Chapitre 3

Méthodes et approches d'alignement d'ontologies

Ce chapitre mettra l'accent sur les approches et les systèmes d'alignement les plus cités dans la littérature. Le but de cette étude est de mettre en lumière les différentes méthodes de rapprochement de la notion de similarité entre les entités d'ontologies ainsi que les techniques sous-jacentes. Certaines des approches recensées sont destinées à l'alignement d'ontologies, d'autres à l'alignement de taxonomies XML ou de schémas de base de données [Euzenat et al., 2004a].
Elles font usage à des techniques différentes comme l'apprentissage machine, l'analyse formelle de concepts, l'analyse statistique etc.

3.1 Méthodes d'alignement

Les méthodes d'alignement peuvent être classifiées en deux catégories :

- Les méthodes locales : ces méthodes s'intéressent à calculer la similarité entre les couples d'entités au niveau local. Ainsi, le calcul de similarité se base sur la définition d'une mesure de similarité entre les couples d'entités en exploitant leurs caractéristiques sans tenir en considération l'aspect global de l'ontologie. A titre d'exemple, calculer la similarité entre les couples d'entités deux à deux en se basant sur le calcul de la similarité lexicale de leurs noms.
- Les méthodes globales : ces méthodes en revanche fonctionnent à l'échelle de l'ontologie. En effet, pour produire des alignements, ces méthodes utilisent les similarités calculées par les méthodes locales en les combinant moyennant d'heuristiques, de fonctions d'agrégation, etc.

3.1.1 Définitions de la similarité et d'autres mesures

Il existe plusieurs moyens d'évaluer la similarité entre deux entités. La technique la plus répandue consiste à définir une formulation mathématique pour évaluer le degré de similarité.

Définition 1 : un indice de similarité [Euzenat et al., 2004a] $\sigma : \mathcal{O} \times \mathcal{O} \rightarrow \mathbb{R}$ est une fonction qui lie un couple d'entités à une valeur réelle qui reflète la similarité entre deux entités. Cette fonction doit satisfaire les propriétés suivantes :

- la positivité :

$$\forall x, y \in \mathcal{O}, \sigma(x, y) \geq 0$$

- la maximalité :

$$\forall x, y, z \in \mathcal{O}, \sigma(x, x) \geq \sigma(y, z)$$

- la symétrie :

$$\forall x, y \in \mathcal{O}, \sigma(x, y) = \sigma(y, x)$$

3.1.2 Méthodes locales

L'opération d'alignement consiste à déterminer pour une entité une autre qui lui correspond le mieux. Ceci revient à définir une mesure qui calcule la similarité entre les couples d'entités. En d'autres termes, une mesure qui minimise la dissimilarité ou inversement maximise la similarité.
Les techniques pour calculer cette mesure trouvent leur existence dans des domaines multiples comme

l'analyse de données, l'apprentissage machine, traitement de langages naturels et bien d'autres.
Les méthodes locales calculent la similarité entre les couples d'entités en exploitant les informations locales du couple comme les URs des entités, leurs structures, etc. Ces méthodes sont à base de la quasi-majorité d'outils d'alignement. En effet, les résultats des méthodes locales sont exploités par les approches globales pour produire des alignements.

3.1.2.1 Méthodes à base de la comparaison lexicale

La comparaison lexicale repose sur l'analyse syntaxique [Cohen et al., 2003] de noms d'entités (URIs) et tout autre élément textuel associé à l'entité (commentaire, URI, etc). Plusieurs mesures ont été proposées pour mesurer la similarité entre les couples d'entités entre autre *l'égalité de chaînes, la distance de Hamming* et la *recherche de sous chaînes.*

3.1.2.2 Méthodes à base de traitement des langages naturels

L'analyse terminologique repose plutôt sur les principes du traitement de langages naturels pour déterminer la ressemblance entre les instances ou entre les classes. La mesure calculée indique le degré de similarité sémantique entre deux noms d'entités. Par conséquent, ces méthodes font souvent recours à l'utilisation des ressources externes comme les thésaurus et les dictionnaires pour construire les listes des synonymes et des hyponymes. Ces listes servent pour determiner la relation entre deux noms [Giunchiglia F. et Yatskevich, 2004, Bouquet et al., 2003]. La bibliothèque de WordNet [Miller, 1995] par exemple implémente diverses mesures de similarité entre deux noms faisant usage à un dictionnaire en anglais.

3.1.2.3 Méthode à base de comparaison structurelle des entités

La comparaison de structures d'entités est une technique à base de plusieurs méthodes d'alignement. La technique peut s'intéresser aussi bien à la comparaison des structures internes qu'aux externes.

La comparaison des structures internes

Les structures internes sont les éléments intrinsèques de l'entité à savoir les attributs, les cardinalités, etc. Les méthodes de comparaison des structures internes [Melnik, 2002] exploitent des critères comme le range des propriétés (attributs et relations), les cardinalités, la transitivité et/ou la symétrie des propriétés d'entités pour calculer la similarité. Les méthodes à base de structures internes sont référencées aussi comme des approches à base de contraintes. Parmi ces méthodes, une approche proposée par

[Li et Clifton, 1994] se base sur la comparison des multiplicités et des propriététés expoitant des caractéristiques d'entités, entre autre l'unicité, les cardinalités, le domaine, l'integrité et les contraintes de sécurité.

La comparaison des structures externes

La comparaison de structures externes d'entités [Shasha et Zhang, 1997, Shasha et al., 2002] consiste quand à elle à comparer leurs contextes existentiels, c'est-à-dire leurs places dans l'ontologie par rapport aux autres entités. L'approche peut par exemple, exploiter les liens d'hiérarchie et/ou de classification entre les entités dans le calcul de similarités. Par exemple, si deux entités se trouvent similaires, il y a une forte probabilité que les nœuds voisins soient similaires aussi. D'une manière générale, les critères pouvant aider à calculer la similarité externe sont :

- Les super entités directes sont déjà similaires [Dieng et Hug, 1998].
- Toutes ou la plupart d'entités sœurs sont déjà similaires.
- Toutes ou la plupart d'entités descendantes sont similaires [Dieng et Hug, 1998].
- Toutes ou la plupart d'entités feuilles sont similaires [Madhavan et al., 2001].
- Toutes ou la plupart d'entités composant les chemins menant aux deux entités d'origines sont similaires [Bach et al., 2004].

Les mesures tirées à base de ces critères sont agrégées par des fonctions pour déduire la similarité entre les entités. Cependant, l'approche en question faillit lorsque les deux domaines représentés par les ontologies en question sont disjoints.

3.1.2.4 Méthodes sémantiques

Les méthodes sémantiques [Euzenat et al., 2004a] disposent d'un modèle théorique utilisé pour justifier leur résultats. Par conséquent, elle sont considérées des méthodes déductives. Des exemples de ces méthodes sont la *satisfaisabilité propositionnelle* (Propositional satisfiability SAT) et les techniques à base de raisonnment dans les logiques de descriptions.

L'approche pour appliquer les techniques SAT dans l'alignement consiste à traduire les requêtes d'appariement à une formule propositionnelle stipulant la subsomption (implication entre les descriptions des entités), puis vérifier sa validité [Giunchiglia F. et Yatskevich, 2004]. Les techniques de logiques de descriptions reposent sur les tests de subsomption pour établir des relations (sous classement ou équivalance) entre les classes dans un sens strictement sémantique. En effet, les techniques procèdent par une fusion de deux ontologies après renommage de classes. Ensuite, elles testent pour chaque paire d'entités la satisfaisabilité d'une assertion affirmant la subsomption entre ces entités.

Il faut signaler que les techniques sémantiques ne sont pas très efficaces lorsqu'elles sont utilisées seules.

En effet, elles requièrent une phase de pre-traitement initiale, souvent une analyse lexicale ou terminologique, pour déterminer les couples d'entités similaires de départ.

3.1.2.5 Méthodes extensionnelles

Les méthodes extensionnelles [Doan et al., 2002] reposent sur la comparaison des extensions de classes (les instances) au lieu de leurs interprétations.

La comparaison des instances en commun : une façon de calculer la similarité entre deux classes A et B partageant des instances en commun est de calculer l'intersection de leurs ensembles d'instances I_A et I_B respectivement sous la considération suivante : les deux classes sont similaires lorsque $I_A \cap I_B = I_A = I_B$ ou plus généralement $I_A \cap I_B = I_B$ ou $I_A \cap I_B = I_A$. De ceci découle une dissimilarité égale à 1 lorsqu'aucune des considérations s'applique (exemple de deux classes qui sont complètement disjointes). Deux mesures en découlent :

- ***Définition 1 : La différence symétrique***
  La différence symétrique, parfois nommée distance de *Hamming*, entre deux ensembles est la fonction de dissimilarité $\delta : 2^E \times 2^E \rightarrow \mathbb{R}$ tel que $\forall x, y \subseteq E. \delta(x, y) = \frac{|x \cup y - x \cap y|}{|x \cup y|}$

- ***Définition 2 : La similarité de Jaccard***.
  La similarité de Jaccard est définie comme suit.

$$\sigma(A, B) = \frac{P(A \cap B)}{P(A, B) + P(\bar{A}, B) + P(A, \bar{B})} \tag{3.1}$$

La mesure est égale à 1 lorsque $A \cap B = \emptyset$ et 1 lorsque $A = B$. $P(A, B)$, $P(A, B)$, $P(\bar{A}, B)$ et $P(A, \bar{B})$ sont des distributions de probabilités.

3.1.3 Méthodes globales

Les méthodes locales définies des mesures de calcul de similarité entre couples d'entités. Les mesures ainsi obtenues sont exploitées par des algorithmes pour produire des alignements. Toutefois, certains traitements globaux sont exigés :

- Agréger les résultats des mesures locales pour calculer la similarité entre les entités composées,
- Développer une stratégie pour résoudre le problème induit par la définition circulaire des concepts et la non linéarité des formules d'agrégation des mesures locales,
- Organiser la combinaison entre les mesures de similarité et les algorithmes à base de ces mesures,
- Impliquer l'utilisateur dans le processus d'alignement,
- Extraire et produire des schémas d'alignement différents.

3.1.3.1 Similarité composée

La similarité composée se base sur l'agrégation des mesures locales. En effet, la nature composite de certaines entités fait dépendre leur similarité des similarités de leurs composants. A titre d'exemple, la similarité d'une classe *A* par rapport à une autre classe *B* dépend naturellement de la similarité de leurs attributs, de leurs super classes, etc. Ainsi, il revient pour calculer la similarite du couple de classes *(A,B)* à composer les similarités des attributs, des super classes et tout autre élément en commun. La composition -ou l'agrégation- se fait moyenant des fonctions ou des heuristiques.

3.1.3.2 Calcul global de la similarité

Dans ce type de méthodes, le calcul de similarité est considéré toujours local. Cependant, la similarité locale définie au niveau d'entités peut conduire à des circularités. Auquel cas, il n'est plus possible de la calculer de la façon directe. En effet, le calcul de similarité est plutôt vu comme un problème d'optimisation globale. Parmi ces méthodes globales nous pouvons situer : *le graphe de propagation de similarité* [Melnik, 2002], *les systèmes d'équations à point fixe* [Euzenat et Valtchev, 2004], *les méthodes d'apprentissage machine* [Doan et al., 2002] et *les réseaux de neurones* [Li et Clifton, 1994].

Méthodes d'apprentissage

Les méthodes d'apprentissage s'avèrent assez utiles dans le domaine d'alignement d'ontologies. Ainsi, dans le but d'aboutir à une similarité globale, les méthodes d'apprentissage exploitent les techniques comme l'analyse formelle de concepts [Stumme et Maedche, 2001], les méthodes bayésiennes [Berlin et Motro, 2002] et les réseaux de neurones [Li et Clifton, 1994].

Un autre domaine d'apprentissage machine dont l'utilisation touche le champ l'alignement d'ontologies, celui des noyaux structurés[1]. Vu leur utilisation dans beaucoup domaines comme la bio-informatique (analyse de la structure d'ADN[Jaakkola et al., 1999]. etc), les noyeaux structurés peuvent être facilement adaptés au contexte d'alignement d'ontologies pour déterminer des fonctions -ou des modèles- pertinents de calcul de similarité. En général, et pour un rapprochement avec notre champ d'étude, un noyau $K(x, x)$ est une fonction qui mesure la similarité entre deux entités x, x' à base de leurs éléments comme les URIs, nœuds voisins, etc. Un bon noyau doit vérifié deux conditions à savoir la *symétrie*[2] et *"positive-semidefinite"*[3].

[1] http ://www.learning-with-kernels.org/

[2] Une fonction binaire $\mathcal{K}(.,.)$ est dite symétrique, si $\forall x, y \in X, K(x, y) = K(y, x)$.

[3] Une fonction binaire $K(.,.)$ est "positive-semidefinite", si $\forall (x_1, x_2), ..., x_n \in X$ la matrice $n \times n$ $(K(x_i, x_j))_{ij}$ est "positive-semidefinite".

Méthode de composition

Dans le cas de ces méthodes [Euzenat et al., 2004a], les valeurs de similarité fournies par les méthodes locales doivent être agrégées. Les méthodes de calcul de similarité et d'évaluation d'alignement sont aussi intégrées par la méthode afin de composer un algorithme d'alignement bien particulier : A titre d'exemple, on peut calculer la similarité lexicale de noms de classes, ensuite la similarité des propriétés puis exécuter une heuristique pour calculer les similarités interdépendantes. Il existe trois manières de composer ces méthodes :

- *composition intégrée* : l'enchaînement des méthodes fait partie de la logique de l'algorithme et indépendant du type d'entrées de l'algorithme. Ce type de composition est adopté par la majorité d'algorithmes d'alignement.
- *composition opportuniste* : il s'agit du cas où le choix de la méthode dépend du type de données d'entrées (taxonomies XML, schéma base de données, etc).
- *composition guidée* : elle est utilisée dans un environnement où le choix de la fonction est effectué par l'utilisateur.

3.2 Quelques exemples de systèmes d'alignement

La littérature fait référence à des systèmes d'alignement multiples [Euzenat et al., 2004a]. Nous présentons quelques cas de systèmes pour mettre en évidence les différentes approches d'alignement et les techniques sous-jacentes.

3.2.1 Le système Glue

Glue [Doan et al., 2002] est une extension du système LSD [Doan et al., 2001] conçu pour l'alignement de schémas de base de données. A l'instar de son prédécesseur, *Glue* combine des techniques d'apprentissage pour produire un alignement entre deux ontologies. Ces dernières peuvent être décrites sous n'importe quel formalisme ou langage de description d'ontologies. L'approche de *Glue* applique les techniques d'apprentissage machine et l'analyse statistique. Le calcul de la similarité entre les couples de classes se base sur la mesure de *Jaccard* 3.2 calculée à l'aide des algortihems d'apprentissage, dits aussi "apprenants", entraînés sur les instances de classes.

$$Jaccard - Sim(A, B) = \frac{P(A, B)}{P(A, B) + P(\bar{A}, B) + P(A, \bar{B})} \tag{3.2}$$

ou $\bar{A}$ est le complément de A. L'idée clée de la mesure de *Jaccard* est que deux classes A et B

sont similaires si leurs instances sont similaires aussi. Afin de calculer la similarité de *Jaccard*, les distributions de probabilités $P(A, B)$, $P(\bar{A}, B)$ et $P(A, \bar{B})$ devraient être calculées. L'architecture de *Glue* s'articule sur trois modules :

1. **Estimateur de distribution** : il applique les techniques d'apprentissage machines sur les instances d'ontologies pour calculer les probabilités dûment mentionées à l'aide d'un ensemble d'apprenants et un méta-apprenant.
2. **Estimateur de la similarité** : cet estimateur produit une matrice de similarités en calculant pour chaque couple de classes leur similarité selon la formule 3.2 et à base de probabilités fournies par le premier estimateur.
3. **Relaxeur de libellés** : à base de la matrice produite par l'estimateur de similarité, le relaxeur de libellés exploite les contraintes spécifiques au domaine et les connaissances heuristiques pour trouver le meilleur appariement.

Observations

1. L'approche d'alignement de *Glue* se base totalement sur l'apprentissage machine et l'analyse d'instances. Toutefois, les autres aspects comme la structure de l'ontologie et les relations entre les entités ne sont pas exploités.
2. La qualité d'alignement est tributaire à la qualité d'apprenants dont la précision est liée à son tour à la représentativité et le volume d'instances de concepts présents dans l'ontologie.
3. L'approche ne tire pas profit des éléments structurels comme les relations entre les entités, etc. Ceci la rend mal adaptée à des ontologies décrites dans des langages formels de description d'ontologies comme OWL.

3.2.2 L'algorithme Cupid

Cupid [Madhavan et al., 2001] est un algorithme qui combine des techniques variées pour identifier les concordances entre les éléments de deux schémas. Il se base sur l'analyse syntaxique de noms d'éléments, leurs types de données, les contraintes, et la structure du schéma. L'algorithme a l'avantage de toucher l'aspect sémantique dans la recherche de similarité entre les concepts. En effet, il cherche des similitudes en exploitant les informations structurelles et les types de données.
A priori, *Cupid* est présenté comme un algorithme générique ; les schémas ou les taxonomies peuvent être exprimées dans n'importe quel langage de formalisation ou de description de connaissances.

L'algorithme procède en trois étapes :

1. **calcul de similarité linguistique** : il consiste à calculer les similarités lexicales *lSim* entre les noms de concepts à l'aide de la normalisation morphologique, la catégorisation par des recherches dans le thesaurus et l'analyse lexicale.
2. **recherche des concordances structurelles** : les deux schémas sont transformées en deux graphes[4] L'algorithme nomée *treeMatch* est utilisé pour calculer les similarités structurelles *Ssim* entre les éléments de deux schémas.
3. **production d'alignement** : il s'agit de calculer des similarités globales (voir la formule 3.3) par une pondération des similarités linguistiques et structurelles et retenir parmi les paires de concepts ceux ayant la similarité supérieure au seuil.

$$Wsim = Wstr * Ssim + (1 - Wstr) * lSim \tag{3.3}$$

lSim est la similitude linguistique, *Ssim* est la similarité structurelle et *Wstr* est un cœfficient dont la valeur est dans [0,1].

Cupid exploite aussi les contraintes préférentielles, présentes dans beaucoup de taxonomies, pour déduire des similarités structurelles (Exemple : les colonnes référencées par la même clé dans deux jointures de vues dans le cas de bases de données).

Observations

- L'algorithme est plutôt destiné à l'alignement de taxonomies sous forme d'arbres. Des taxonomies comme les graphes doivent être transformées au préalable sous forme d'arbres. Cette transformation ne preserve pas toujours les éléments des taxonomies en question et par conséquent pourraient avoir un impact sur la qualité de l'alignement produit.

3.2.3 La méthode FCA-Merge

La méthode *FCA-Merge* [Stumme et Maedche, 2001] est une approche buttom-up qui applique les techniques de traitement du langage naturel et l'analyse formelle de concepts pour fusionner deux ontologies partageant les mêmes ensembles d'instances. La démarche peut être résumée comme suit :

1. L'extraction d'instances : elle se fait par une selection d'un ensemble de documents spécifiques au domaine d'intérêt et couvrant tous les concepts de deux ontologies en question.

[4]Cette transformation permet à partir de deux arbres, d'avoir deux graphes pour pouvoir appliquer l'algorithme treeMatch.

2. L'analyse linguistique : elle s'effectue à l'aide d'un corpus pour analyser le contenu de ces documents afin de produire une matrice binaire [D,C]. Chaque élément de la matrice indique l'existence d'une relation entre le concept C_i et le document D_j.
3. La génération de treillis : elle s'appuie sur les techniques de l'analyse formelle de concepts. Les deux treillis générés représentent les deux ontologies originales.
4. La fusion : l'algorithme reçoit les deux treillis de concepts, les désambiguïse en supprimant les concepts redondants, et produit un autre treillis -la fusion de deux treillis originaux-.
5. Le filtrage semi-automatique de concepts. Cette opération requière l'intervention d'un expert pour le choix de concepts les plus pertinents au domaine. Ces derniers représentent les concepts de l'ontologie fusionnée.

Observations

- La méthode quoi qu'elle se base sur des notions formelles de modélisation de concepts, ne repose pas sur des techniques aussi robustes pour le choix de documents couvrant les deux ontologies.
- La génération de deux treillis se base sur la détermination des relations entre les documents et les concepts de l'ontologie. Cette opération dépend totalement de l'analyse linguistique de documents,
- Le choix de documents couvrant le plus largement possible les concepts de deux ontologies à fusionner reste une tache manuelle très subjective,
- L'approche, plutôt destinée à la fusion d'ontologies, peut être capitalisée pour produire des alignements en considérant les concepts retenus de la fusion comme ceux les plus semblables.

3.2.4 L'algorithme Anchor-PROMPT

Anchor-PROMPT [Noy et Musen, 2001] (extension de *PROMPT* connu aussi sous le nom *SMART*) est présenté comme un algorithme semi-automatique pour l'alignement et la fusion d'ontologies. Il se base sur l'analyse terminologique, l'analyse de la structure interne et externe et sur l'analyse taxonomique d'entités. Le processus d'alignement commence par une analyse lexicale de noms d'entités pour construire une liste de concepts similaires dits entités-ancres. Ensuite, l'algorithme construit un graphe libellé par ontologie dont les nœuds sont les classes de l'ontologie et les arcs sont les relations entre les classes. L'algorithme reçoit la liste des couples d'entités à aligner et effectue une analyse des chemins des sous-graphes délimités par les couples-ancres. Par cette analyse, l'algorithme calcule le nombre d'occurrences qu'un concept apparaît dans la même position dans les mêmes chemins. A base de ce nombre, l'algorithme détermine les concepts similaires.

Observations

- Anchor-Prompt n'affiche pas des résulats aussi bons si une ontologie est plus profonde que l'autre. Dans telle situation, les chemins de graphes -constitués à partir des couples de concepts départ- sont non équilibrés. Ceci à un impact négatif sur le résultat de similarité caluclée.
- Il affiche des résultat meilleures lorsque les ontologies sont équilibrées, contruites suivant une approche similaire et selon un même modèle de représentation de connaissance,
- Anchor-Prompt requière un minimaum de couples de concepts similaires déterminés par l'utilisateur tout au début.
- Vu ces performances, l'algorithme a été integré -comme un plug-in- dans *protégé-2000*[5].

3.2.5 Le système COMA

Le système *COMA* (COmbination of MAtching algorithms) [Do et Rahm, 2002] est un outil d'appariement générique qui implémente différents apparieurs. Il fournit une bibliothèque extensible d'algorithmes d'alignement : c'est une plate-forme pour combiner les résultats obtenus et évaluer les performances d'apparieurs. La bibliothèque contient six apparieurs : cinq sont hybrides et un autre dit "orienté-réutilisation". La plupart des apparieurs implémentent les techniques d'analyse syntaxique. Les autres implémentent de techniques inspirées du *Cupid*. L'idée originale de *COMA* vient avec l'apparieur "orienté-réutilisation". En effet, ce dernier réutilise les résultats obtenus précédemment pour apparier de nouveaux schémas. Ces derniers sont encodés sous forme de graphes orientés et acycliques dont les éléments sont les chemins. Dans *COMA* le processus d'appariement pourrait être effectué par itérations et en intéraction avec l'utilisateur qui approuve les ressemblances ou les discordances des couples proposées par le système.

Observations

- *COMA* a le mérite de s'exceller comparativement à d'autres méthodes comme *Cupid* et *Glue* selon l'étude [Doan et al., 2003].

3.2.6 L'algorithme Versatile-Graph-Flooding Similarity (VGFS)

VGFS [Melnik, 2002] est un algorithme générique qui utilise le calcul à point fixe pour déterminer les correspondances entre les nœuds de deux graphes représentant les deux ontologies d'origine à aligner. L'algorithme repose sur l'assomption de la propagation de similarité à travers les nœuds d'un graphe. En

[5]protégé est une platefrom de gestion de connaissances http ://protege.stanford.edu/

effet, la similarité des couples se propage et fait augmenter celle des couples adjacents. La démarche générale de l'algorithme est comme suit :

1. Représenter les deux ontologies, chacune sous forme d'un graphe suivant les spécifications du modèle OIM [Shutt et al., 1999] (open Information Model).
2. Calculer le premier appariement entre les nœuds de deux graphes à base de calcul de similarité lexicale de leurs identifiants.
3. Produire le graphe de connectivité, la fusion de deux graphes, pour en construire un autre dit graphe de propagation de similarité. Les nœuds de ce dernier sont les couples de concepts à aligner. Les arcs déterminent le sens de la propagation de la similarité et sont pondérés par des cœfficients compris dans $[0, 1]$. Chaque cœfficient représente la contribution du couple dans le calcul de la similarité du couple ascendant.
4. Calculer la similarité par une fonction d'agrégation linéaire à point fixe ε déterminé par l'utilisateur. Le processus de calcul est itératif et s'arrête lorsque la différence de similarité de deux itérations successives pour tous les couples est inférieure à ε.
5. Filtrer les couples ayant la plus grande similarité pour produire un appariement.

Observations

- Les transformations apportées pour produire le graphe final, ne sont pas adaptées aux cas de taxonomies ayant un nombre important d'individus. Dans tels cas, les graphes résultats sont complexes. La complexicité de calcul et le temps de traitement deviennent considérables,
- L'idée de propagation de similarité a affiché des résultats intéréssants,
- L'algortihme a manifesté aussi un degré important d'indépendance par rapport au modèle ou au langage de représentation d'ontologies.

3.2.7 Tableau récapitulatif des méthodes

La table[6] TAB. 3.1 résume les méthodes et approches d'alignement que nous avons présentées.

3.3 Conclusion

Dans ce chapitre, nous avons mis en exergue quelques méthodes d'alignement d'ontologies les plus connues dans la littérature. Ainsi, nous avons montré pour chacune les techniques de rapprochement

[6]Une partie de la table a été tirée du rapprot éléboré par le Knowldge Group [Euzenat et al., 2004a].

MÉTHODE	T	TS	TL	I	S	ST	E	SM	SU
FCA-Merge					X		X		
Cupid	X	X	X	X	X				
Anchor-Prompt	X			X	X	X			X
Glue							X		
COMA	X	X			X				X
VGFS	X		X		X				

TAB. 3.1 – Table des méthodes d'alignement présentées et les approches de similarités adoptées. T : Terminologique, TS : Terminologique à base de chaine de caractères, TL : Teminologique à base de Langage, I : Internes, S : Semantique, ST : Semantique à base Terminologique, E : Extentionelle, SM : Semantique à base d'un Modèle et SU : Semantique à base de choix de l'Utilisateur

de calcul de la similarité utilisées. En effet, les méthodes présentées ont recours à des techniques diverses capitalisant les travaux effectués dans des domaines de recherches multiples comme l'analyse lexicale et sémantique, l'analyse statistique, les techniques d'apprentissage machine etc. Cependant, peu de méthodes focalisent sur la question de rapprochement de la similarité à travers un modèle englobant tout le potentiel informationnel existant de l'ontologie. Ce fait, pourrait être expliqué d'une part par la diversité des domaines de recherche s'intéressant à la question d'alignement : chacun traite la question par rapport à sa propre vision et utilisant ses propres techniques (l'apprentissage machine versus l'analyse formelle de concepts par exemple) et d'une autre part par la nature difficile du problème lui même. Ce dernier touche à la question de modélisation et de formulation de la similarité en exploitant l'aspect symantique des concepts au sein de l'ontologie.

Dans le chapitre suivant, nous présentons notre méthode d'alignement d'ontologies OWL. Elle s'inspire de l'approche de l'algorithme Verstile-Graph-Flooding plus particulièrement de l'idée de propagation de la similarité à travers les nœuds du graphe. En outre, notre méthode utilise une représentation plus simple et plus parlante d'une ontologie : le graphe, par ces éléments structuraux, traduit fidèlement les différents éléments du langage OWL-Lite (classes, propriétés, cardinalités, etc). Le modèle de calcul de similarité à base du graphe inclut en l'occurence ces éléments en vue d'aboutir à un rapprochement efficace de la similarité entre les entités d'ontologies.

Chapitre 4

Notre approche d'alignement d'ontologies OWL-Lite

Dans ce chapitre, nous présentons au lecteur notre approche d'alignement d'ontologies *OWL-Lite*. Elle s'inspire des techniques et études réalisées sur l'alignement d'ontologies [Euzenat et Valtchev, 2003, Melnik, 2002] et les travaux de recherche effectués sur la comparaison d'objets [Valtchev, 1999] dans une base de connaissances. L'approche prend en considération toutes les caractéristiques de l'ontologie à savoir la similarité terminologique, structurelle et extensionnelle d'entités d'ontologies. En outre, elle propose un modèle de calcul de similarité qui résout le cas des circularités entre les concepts lors du processus d'alignement.

Le chapitre s'organise comme suit : nous commençons par les motivations de cette recherche et les défis rencontrés. Ensuite, nous abordons les grandes lignes de l'approche : nous expliquons le processus de construction d'OL-Graph, le modèle de calcul de similarité et le processus de calcul de similarité entre les couples d'entités. Un exemple illustratif de calcul est donnée à titre d'application de l'approche. Finalement, nous parlons de l'étape de la production d'alignement en expliquant la méthode de production d'alignement et le schéma adopté.

4.1 Motivations

Il s'agit dans un premier plan de décrire une ontologie *OWL-Lite* sous forme d'une rprésentation dite OL-Graph qui reflète toute l'information structurelle, textuelle et relationnelle de l'ontologie. Ainsi à base de cette représentation nous aimerions dans un second lieu l'utiliser pour mettre en œuvre un outil automatique d'alignement d'ontologies *OWL-Lite*, experimenter et conclure autour de notre approche d'alignement afin de l'évaluer et de juger de ses performances.

Un autre objectif qui découlera à la lumière des résultats obetnus, est de généraliser notre approche eventuellement sur les autres éléments du langage OWL (les langages DL et Full).

4.2 Défis rencontrés

L'approche d'alignement se heurte à deux défis principaux : le premier est lié à la grammaire du langage OWL-Lite et le deuxième à la circularité posée par l'interdépendance entre les définitions de concepts.

4.2.1 Grammaire du langage OWL

Quoi que le langage OWL dans sa version *OWL-Lite* ne soit pas trop expressif, sa grammaire pose néanmoins des défis notables. On cite principalement :

- La présence d'entités anonymes (comme les classes et les individus anonymes) : ce genre d'entités contribuent à l'enrichissement de la description des autres entités sans nécessairement avoir leurs propres identifiants (URIs). Ceci engendre certaines difficultés lors de leur représentation par les éléments structuraux de l'OL-Graph. En effet, pour mettre le doit sur l'une de ces difficultés, nous considerons l'exemple d'une classe anonyme decrite comme une collection d'individus. La représentation de cette classe, par un nœud de type classe, pose un problème de nomage de ce nœud (l'attribution d'un identifiant). La question est cruciale, sachant que les identifiants (URIs) des éléments de l'OL-Graph contribuent dans le calcul de similarité des nœuds entre les ontologies et par conséquent sur le résultat de l'alignement produit.
- Le traitement de l'information implicite ou inférée : par le mécanisme d'héritage le langage *OWL-Lite* propage de l'information des entités ascendantes vers les entités descendantes (exemple de sous classement entre les classes, les propriétés).
- L'interprétation des constructeurs d'OWL : certains de ces constructeurs ont la même sémantique alors qu'ils sont syntaxiquement différents comme les constructeurs *subClassOf* et *intersectionOf*.

- Les descriptions ayant des différences sémantiques subtiles. Ceci fait surgir un problème quand à leur interprétation adéquate par les structures d'OL-Graph.

4.2.2 Problème de la circularité

Le problème de la circularité dans le calcul de la similarité est un problème bien connu. En effet, la description des concepts en fonction d'autres peut crée des circularités. Étant donnée que le calcul de la similarité d'un couple dépend des similarités d'entités voisines, dans le cas de circularité, le calcul n'est pas direct. Dans la section 4.3.4.1 nous abordons le problème plus en détail.

4.3 Grandes lignes de l'approche

L'approche s'articule sur trois grandes lignes :

- Proposer une représentation de l'ontologie décrite en *OWL-Lite* sous forme d'un graphe OL-Graph. Le graphe représente un "meta-modèle" qui reflète toutes les entités, les relations et les instances d'ontologies *OWL-Lite*.
- Définir une mesure de calcul de similarité entre les entités d'ontologies par catégorie d'entité.
- Définir un modèle global de calcul de similarité tout en remédiant au problème de la circularité.

4.3.1 Modèle du graphe OL-Graph

Il fallait tout d'abord penser à un modèle de graphe générique et adéquat pour représenter une ontologie OWL. Le rôle d'un tel graphe est de refléter toutes les entités, les relations et les individus de l'ontologie décrite sous le langage *OWL-Lite*. En effet, un tel besoin va au delà des questions algorithmiques : il s'agit principalement de transformer une ontologie OWL en une structure (graphe) qui conserve et met en evidence toute l'information contenue dans une ontologie *OWL-Lite*. Ceci permet d'avoir comme entrée deux graphes représentant chacun une ontologie. Ainsi, des techniques de calcul de similarité pourront être appliquées éventuellement pour mesurer le degré de ressemblance entre les entités à base de ces deux graphes. Par conséquent, la capacité du graphe à représenter toute l'information exprimée dans une ontologie aura un impact aussi bien sur les résultats de calcul de similarité que sur la fiabilité des autres outils d'alignement et de visualisation qui y sont basés.

ENTITÉ OWL	NŒUD DANS OL-GRAPH
Classe	ClassNode
Individu	ObjectNode
Type de donnée	DatatypeNode
Valeur de donnée	DataValueNode
Propriété de nature "objet"	ObjectPropertyNode
Propriété de nature "type de donnée"	DatatypePropertyNode
Relation stand-alone de nature "objet"	DatatypeRelationNode
Relation stand-alone de nature "type de donnée"	ObjectRelationNode
Instance de propriété de nature "objet"	ObjectPropertyInstanceNode
Instance de propriété de nature "type de donnée"	DatatypePropertyInstanceNode

TAB. 4.1 – Table des entités OWL et leurs équivalents dans OL-Graph

4.3.1.1 Ontologie OWL vue comme un OL-Graph

Pour répondre à ces questions, une ontologie *OWL-Lite* est perçue comme un graphe que nous appelons OL-Graph représenté par la figure FIG. 4.1. Ce graphe constitue un modèle de représentation d'une ontologie décrite sous le langage *OWL-Lite*. Les nœuds du graphe représentent les entités de l'ontologie comme les classes, les objets, les propriétés, etc. Les arcs décrivent en revanche les relations exprimées au sein de l'ontologie comme les relations de spécialisation, d'instantiation, etc.

4.3.1.2 Catégories de nœuds d'OL-Graph

Les catégories des nœuds d'OL-Graph représentent à priori les types d'entités qu'on trouve dans une ontologie *OWL-Lite*. La table TAB. 4.1 donne la liste d'entités OWL et les noms de nœuds équivalents dans l'OL-Graph.

A noter que les nœuds des *relations stand-alone* de nature "objet" et de "type de donnée" correspondent à des relations génériques d'où les propriétés et les instances de propriétés sont dérivées. Les relations stand-alone sont des entités propres à OL-Graph[1].

[1]Il n'existe pas dans le langage OWL-lite des constructeurs pour déclarer des relation stand-alone.

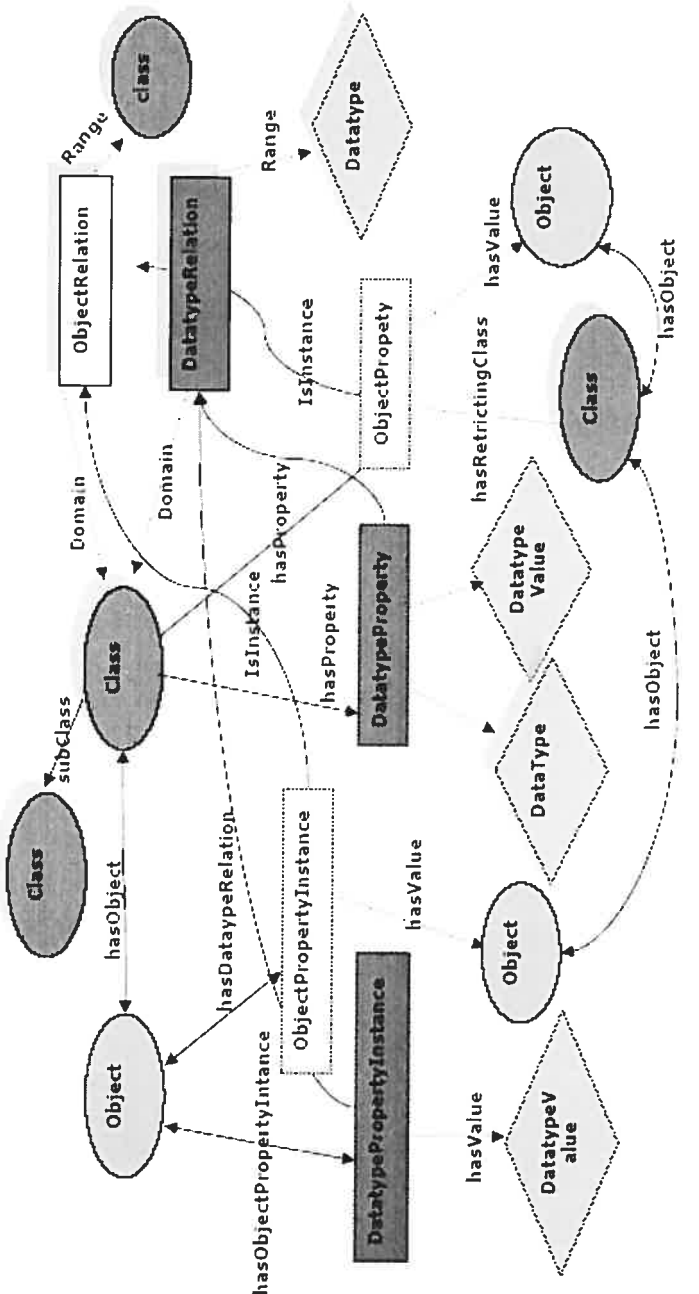

FIG. 4.1 – Structure d'OL-Graph

LIEN	SYMBÔLE	CONTEXTE
Lien de spécialisation	$\mathcal{S}$	entre deux classes ou deux propriétés
Lien d'instantiation	$\mathcal{I}$	entre objets et classes, instances de propriétés et propriétés, valeurs et "type de donnée"
Lien d'attribution	$\mathcal{A}$	entre classes et propriétés, objets et instances de propriétés
Lien de valuation	$\mathcal{U}$	valuation d'une propriété ou instance de propriété de nature "objet"

TAB. 4.2 – Table des relations OWL et leurs équivalents dans OL-Graph

4.3.1.3 Liens entre les nœuds d'OL-Graph

Les liens entre les nœuds d'OL-Graph traduisent les relations sémantiques entre les entités d'ontologie. La table TAB. 4.2 présente des liens recensés dans l'ontologie et leurs équivalents dans l'OL-Graph.

4.3.1.4 Construction d'OL-Graph

Une ontologie décrite en OWL-Lite est transformée en un OL-Graph suivant le processus suivant :

- **construction** : elle consiste à "parser" l'ontologie décrite en langage *OWL-Lite*, dégager les entités et les relations de l'ontologie, et construire leurs équivalents en terme de nœuds et d'arcs dans l'OL-Graph. Ainsi, les nœuds et les arcs représentent respectivement les entités et les relations qui existent dans une ontologie. A titre d'exemple, une classe sera représentée par un nœud de type *ClasseNode* et sera connectée aux nœuds représentant ces super classes, ces propriétés, et ces instances par les arcs reflétant respectivement les relations de spécialisation, d'attribution et d'instantiation.

- **aplatissement** : lors du processus de construction initiale du OL-Graph, des nœuds auxiliaires sont ajoutés au OL-Graph. Ces nœuds aident uniquement à construire toute la structure d'une entité lors de son "parsing". Ainsi, l'opération d'aplatissement consiste à supprimer, parmi les nœuds ajoutés, les nœuds superflus tout en préservant la définition de l'entité telle qu'elle est exprimée par la syntaxe OWL dans l'ontologie.
- **complétion** : la complétion d'OL-Graph consiste à enrichir la structure d'OL-Graph par l'ajout de

liens et de nœuds. Le but de la complétion est de prendre en considération l'information implicite propagée par l'héritage entre les propriétés, les classes etc ainsi que de faciliter la navigabilité dans le graphe. Cette complétion n'a aucun effet de distorsion de la sémantique des relations et des entités exprimées dans l'ontologie.

La complétion se fait comme suit :

- instantiation des relations stand-alone : il s'agit de dériver les propriétés de nature "objet" ou de "type de donnée" à partir des relations stand-alone correspondantes. Ces propriétés sont attachées soit à des nœuds de classes ou de "type de donnée".
- ajout des liens d'instantiation entre les classes et les individus.
- ajout des liens d'instantiation entre les valeurs et leurs types.
- ajout des liens d'instantiation entre les instances de propriétés et les relations stand-alone.

4.3.2 Exemple d'OL-Graph

La représentaton de l'ontologie OWL-Lite ci-dessous, est représentée par le OWL-Graph FIG.4.2. Nous nous sommes limités aussi à une ontologie de petite taille pour mieux saisir la structure de OL-Graph. A noter que les arcs ont été libéllés pour déterminer la nature des relations entre les nœuds (concepts de l'ontologie)

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE rdf:RDF (View Source for full doctype...)>
<rdf:RDF
   xmlns="http://co4.inrialpes.fr/align/Contest/104/onto.rdf#"
   xmlns:units="http://visus.mit.edu/fontomri/0.01/units.owl#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:ical="http://www.w3.org/2002/12/cal/#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:wot="http://xmlns.com/wot/0.1/"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:dctype="http://purl.org/dc/dcmitype/">

 <owl:Class rdf:ID="Human">
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="#dwelling" />
       <owl:allValuesFrom rdf:resource="#Flat" />
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#spouse" />
         <owl:allValuesFrom rdf:resource="#Human" />
```

```
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
           <owl:onProperty rdf:resource="#spouse" />
           <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1
           </owl:maxCardinality>
         </owl:Restriction>
      </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:ID="Flat">
    <rdfs:label xml:lang="en">Flat</rdfs:label>
      <rdfs:subClassOf>
       <owl:Restriction>
        <owl:onProperty rdf:resource="#rooms" />
        <owl:allValuesFrom rdf:resource="#Room" />
       </owl:Restriction>
     </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:ID="Room">
    <rdfs:label xml:lang="en">Room</rdfs:label>
  </owl:Class>

<owl:DatatypeProperty rdf:ID="name">
  <rdfs:domain rdf:resource="#Human" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  <rdfs:label xml:lang="en">key</rdfs:label>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="age">
  <rdfs:domain rdf:resource="#Human" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger" />
  </owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="address">
  <rdfs:domain rdf:resource="#Flat" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  <rdfs:label xml:lang="en">key</rdfs:label>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="surface">
  <rdfs:domain rdf:resource="#Room" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger" />
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="dwelling">
    <rdfs:domain rdf:resource="#Human" />
    <rdfs:range rdf:resource="#Flat" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="spouse">
    <rdfs:domain rdf:resource="#Human" />
    <rdfs:range rdf:resource="#Human" />
```

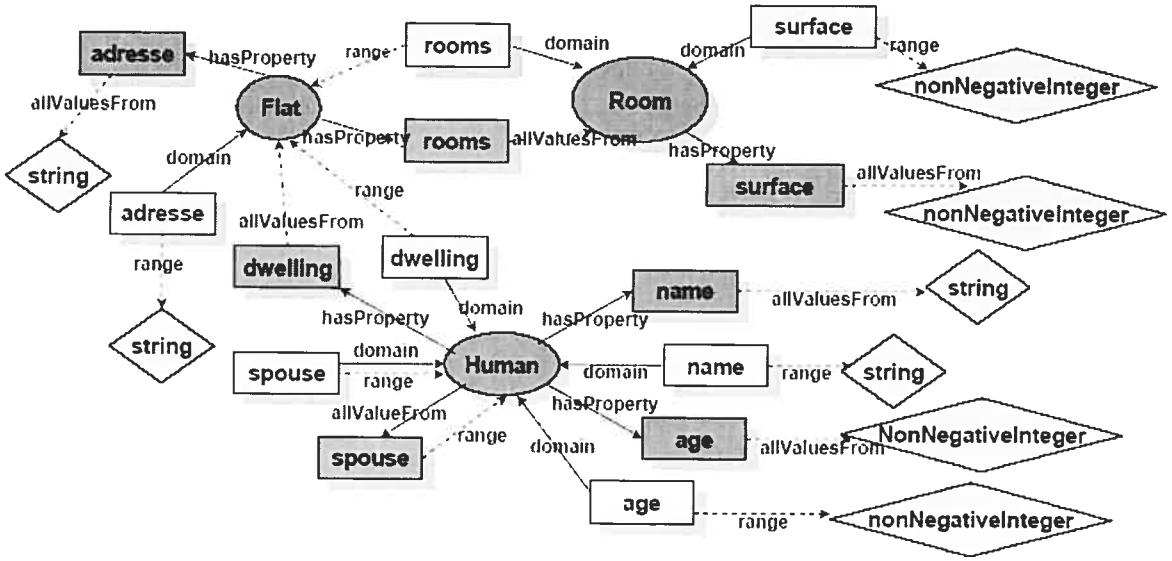

FIG. 4.2 – Ontologie représentée selon OL-Graph

```
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="rooms">
   <rdfs:domain rdf:resource="#Flat" />
   <rdfs:range rdf:resource="#Room" />
</owl:ObjectProperty>
</rdf:RDF>
```

4.3.3 Modèle de calcul de la similarité

Le but par la définition d'une mesure de similarité est de permettre en premier lieu la conception de méthodes automatiques ou semi-automatiques d'alignement. En effet, plusieurs mesures de similarité ont été suggérées. Cependant, la difficulté à proposer des mesures de similarité précises émane du fait que le facteur sémantique est impossible à exprimer par une formule mathématique qui inclut toute l'information sémantique entourant une entité. Par conséquent, les approches d'alignement font souvent recours à des heuristiques combinant plusieurs techniques à la fois pour rapprocher des modèles de calcul de similarité entre les entités d'ontologies.

4.3.3.1 Principe de base de la mesure de similarité

Notre approche de calcul de similarité suit la structure d'OL-Graph dans le calcul de la similarité entre les nœuds d'ontologies. Les nœuds de deux graphes d'entrée ne sont que les entités d'ontologies à aligner. Ainsi, le modèle de calcul assigne pour chaque catégorie de nœuds une fonction d'agrégation. Cette fonction exploite toutes les informations descriptives du couple d'entités à aligner en les englobant dans deux niveaux de calcul : un premier niveau *local*, et un autre *global*.

4.3.3.2 Calcul de la similarité locale

Le calcul de la similarité se fait par catégorie d'entité (classe, objet, propriété, relation...). Ainsi, étant donnée deux ontologies, chacune représentée par un OL-Graph, il s'agit de calculer la similarité entre les couples d'entités pour une même catégorie en exploitant la similarité de leurs descripteurs locaux et celle des entités voisines. On distingue deux ordres de calcul :

1. Le premier ordre consiste à calculer la similarité lexicale/terminologique *simL* entre les descripteurs d'entités comme les URIs, les commentaires, etc. Le calcul de la similarité lexicale s'effectue à base d'algorithmes d'analyse lexicale comme *EditDistance* [Ristad et Yianilos, 1998] *Leveinstein* [Levenshtein, 1966]. Celui de la similarité terminologique se fait par l'API de WordNet.

2. Le deuxième ordre s'intéresse au calcul de la similarité $Sim(Y)$ entre les ensembles de nœuds voisins par catégorie Y.

En effet, les nœuds adjacents d'un couple sont organisés par catégories. Il arrive souvent que le calcul de la similarité d'une catégorie Y revient à calculer la similarité entre deux ensembles d'entités : le cas typique, présenté par la figure FIG. 4.3, est le calcul de similarité d'un couple de classes. Ainsi, le calcul fait appel entre autre au calcul de la similarité de deux ensembles de super classes $S(c)$ et $S(c')$.
Pour résoudre ce cas problématique, l'approche repose sur la définition de la similarité *Match-Based similarity* 4.1 entre deux ensembles [Valtchev, 1999]

$$MSim(S, S') = \frac{\sum_{\langle c,c'\rangle \in Paires(S,S')} Sim_C(c, c')}{max(|S|, |S'|)} \tag{4.1}$$

La formule calcule la similarité entre deux ensembles d'entités. Les arguments de la fonction sont les deux ensembles $S(c)$ et $S'(c')$. Le résultat de la fonction est la moyenne normalisée des meilleures similarités des couples de l'ensemble $\mathcal{P}$ produit de $S \times S'$. Ceci requière que les similarités de ces couples soient déjà calculées.

Le choix des meilleures similarités entre les couples de deux ensembles pourrait être formulé et résolu par une approche gloutonne ou par la programmation dynamique. Cette opération nomée souvent "filtrage", consiste en fait à choisir parmi les couples ceux ayant les plus grandes similarités. La méthode gloutonne se caractérise par une recherche locale des couples ayant la plus grande similarité. Par contre, la programmation dynamique procède via une approche d'optimisation globale. L'objectif final est de produire à base de résultat d'alignement, un appriement entre les concepts d'ontologies alignées.

4.3.3.3 Calcul de la similarité globale

Le calcul de la similarité globale repose sur une agrégation des similarités locales par une fonction quasi-linéaire 4.2 où pour chaque facteur $MSimY$ un poids $\Pi_{\mathcal{F}}^{X}$ lui a été associé. Le poids $\Pi_{\mathcal{X}}^{L}$ correspond au facteur assigné à la similarité lexicale/teminologique. En général, les poids $\Pi_{\mathcal{F}}^{X}$ et $\Pi_{\mathcal{X}}^{L}$, désignés aussi comme des facteurs d'agrégation, sont des valeurs déterminées et attribuées par l'usager à chaque catégorie d'entité.
Formellement, étant donnée une catégorie d'entité X et l'ensemble des relations impliquées, $N(X)$, la mesure de similarité $Sim_X : X^2 \rightarrow [0,1]$ est définit comme suit :

$$Sim_X(x, x') = \sum_{\mathcal{F} \in \mathcal{N}(X)} \Pi_{\mathcal{F}}^{X} MSim_Y(\mathcal{F}(x), \mathcal{F}(x')) \tag{4.2}$$

La fonction est normalisée, c'est-à-dire $\sum(\Pi_{\mathcal{F}X}) = 1$.

La table TAB. 4.4 présente pour chaque catégorie d'entité les catégories d'entités et les facteurs intervenant dans son calcul. Chaque entité (nœud) est identifiée ($\lambda : C \cup O \cup R \cup P \cup D \cup A \rightarrow$ URIRef) par un URI.

4.3.3.4 Exemple : calcul de similarité d'un couple de classes

Pour illustrer le principe général de calcul de la similarité entre les couples d'entités, nous présentons un exemple de calcul de la similarité pour le cas d'un couple de classes.

La figure FIG. 4.3 représente deux classes suivant le modèle d'OL-Graph. Chaque classe est liée à l'ensemble de ces super-classes, à ces objets et à ces propriétés de nature "objet" et de "type de donnée". Le calcul de la similarité $Sim(c, c')$ entre les deux classes est effctué suivant la formule 4.3 par l'agrégation des similarités locales suivantes :

- Sim_L : similarité terminologique des deux classes,
- $MSim_C$: similarité de deux ensembles des super classes $S(c)$ et $S(c')$,

SYMBÔLE	NŒUD REPRÉSENTÉ
C	classe
O	objet
R	relation
P	propriété
D	type de donnée
V	valeur de donnée
A	instance de propriété
L	restriction par libellé

TAB. 4.3 – Table des nœuds d'OL-Graph et leurs symboles

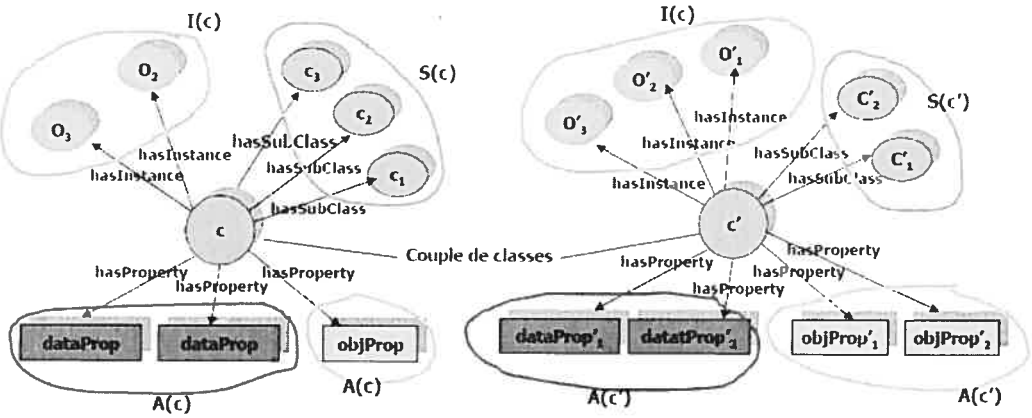

FIG. 4.3 – Représentation d'un couple de classes selon OL-Graph

FONCTION	NŒUD	FACTEUR	MESURE
Sim_O	$o \in O$	$\lambda(o)$	sim_L
		$a \in A, (o,a) \in \mathcal{A}$	$MSim_A$
Sim_A	$a \in A$	$r \in \mathcal{R}$,$(a,r) \in \mathcal{R}$	Sim_R
		$o \in O, (a,o) \in \mathcal{U}$	$MSim_O$
		$v \in$V$, (a.v) \in \mathcal{U}$	$MSim_V$
Sim_V	$v \in V$	valeur	type dépendant
Sim_C	$c \in C$	$\lambda(c)$	sim_L
		$o \in O, (a.o) \in \mathcal{U}$	$MSim_O$
		$p \in P, (c,p) \in \mathcal{A}$	$MSim_P$
		$c' \in C, (c,c') \in \mathcal{S}$	$MSim_C$
sim_D	$d \in D$	$\lambda(r)$	XML-Schema
Sim_R	$r \in R$	$\lambda(r)$	sim_L
		$r \in C, (r, domain, c) \in \mathcal{R}$	$MSim_C$
		$c \in C, (r, range, c) \in \mathcal{R}$	$MSim_C$
		$d \in D, (r, range, d) \in \mathcal{R}$	$MSim_D$
		$r' \in R, (r,r') \in \mathcal{S}$	$MSim_R$
Sim_P	$p \in P$	$r \in \mathcal{R}, (p, r \in \mathcal{S})$	Sim_R
		$c \in C, (p, all, c) \in \mathcal{R}$	$MSim_C$
		$n \in \{0,1,\infty\}, (p, card, n) \in \mathcal{R}$	égalité

TAB. 4.4 – Table des fonctions de similarité et leurs compositions

- $MSim_{P_{dt}}$: similarité de deux ensembles de propriétés de nature "type donnée" $\mathcal{A}(c)$ et $\mathcal{A}(c')$,
- $MSim_{P_o}$: similarité de deux ensembles de propriétés de nature "objet" $\mathcal{A}(c)$ et $\mathcal{A}(c')$,
- $MSim_O$: similarité de deux ensembles d'objets $\mathcal{I}(c)$ et $\mathcal{I}(c')$.

$$Sim_X(c,c') = \Pi^c_L Sim_L(\lambda(c),\lambda(c')) + \Pi^c_S Sim_S(\mathcal{S}(c),\mathcal{S}(c')) \quad (4.3)$$

$$+ \Pi^c_{P_o} Sim_{P_o}(\mathcal{A}(c),\mathcal{A}(c')) + \Pi^c_{P_{dt}} Sim_{P_{dt}}(\mathcal{A}(c),\mathcal{A}(c')) \quad (4.4)$$

$$+ \Pi^c_O Sim_O(\mathcal{I}(c),\mathcal{I}(c')) \quad (4.5)$$

Pour chaque catégorie d'entité Y intervenant dans le calcul de la similarité d'une catégorie X, un poids Π^X_Y est assigné reflétant la contribution de la similarité de la catégorie Y dans le calcul de similarité d'un couple d'une catégorie X. La similarité $MSim_Y$ de deux ensembles d'entités d'une catégorie Y est caclulée suivant la formule 4.1.

4.3.3.5 Adaptation des facteurs d'agrégation

Une des conclusions tirée lors de ce travail porte sur la condition de poids uniformes associés aux similarités Sim_Y par catégorie d'entités [Euzenat et al., 2004b]. En effet, le calcul de la similarité par poids uniformes tend à favoriser les couples d'entités ayant une description complète c'est-à-dire les couples vérifiant la condition suivante : $\forall$ catégorie Y : $P = S \cup S' \neq \emptyset$

Afin d'expliciter ce cas, nous présentons un exemple de calcul de la similarité d'un couple de classes **(Person, Car)** avec et sans uniformisation des poids. La table TAB.4.5 présente les poids associés aux catégories d'entités. La table TAB. 4.6 donne des similarités fictives par catégorie d'entités du couple **(Person, Car)**. La table TAB. 4.7 résume la liste d'entités pour chacune des classes du couple. Nous remarquons que les deux catégories : objet et propriété de nature "type de donnée" ne satisfont pas la condition sus-mentionnée (la classe **Car** n'a pas d'objets, la classe **Person** n'a pas de propriétés de nature "type de donnée")

Selon la formule 4.2, le calcul de la similarité du couple **(Person, Car)** sans uniformisation de poids est comme suit :

$$Sim(Person, Car) = \frac{\sum (0.82 * 0.50), (0.10 * .10), (0.21 * 0.20)}{\sum 0.50, 0.10, 0.10, 0.20, 0.10} = 0.462$$

Avec uniformisation de poids, elle devient :

CATÉGORIE	POIDS
Similarité lexicale	0.50
Propriété de nature "objet"	0.10
Propriété de nature "type de donnée"	0.10
Super classe	0.20
Objet	0.10

TAB. 4.5 – Table de poids associés aux catégories d'entités du couple (Person, Car)

CATÉGORIE	SIMILARITÉ
Similarité lexicale	0.25
Propriété "objet"	0.64
Propriété "type de donnée"	0.00
Super classe	0.21
Objet	0.00

TAB. 4.6 – Table des similarités des catégories d'entités du couple (Person, Car)

CATÉGORIE	PERSON	CAR	PRODUIT P
Propriété "objet"	hasSpouse	hasOwner	{(hasSpouse, hasOwner)}
Propriété "type de donnée"	hasName		{∅}
Super classe	Adult	Engine	{(Adult, Engine)}
Objet	Harry, Mario		{∅}

TAB. 4.7 – Table des entités du couple (Person, Car) par catégorie

$$Sim_C(Person, Car) = \frac{\sum (0.82 * 0.50), (0.10 * 0.10)}{\sum 0.50, 0.10, 0.10, 0.20} = 0.577$$

Nous remarquons une différence de similarité de 0.115. Cette perte de similarité ($\simeq 11\%$) n'est pas négligeable du moment qu'elle se répercute directement et indirectement, par le mécanisme de récursivité, sur les similarités des autres couples d'entités. Dans l'exemple, la perte de similarité aura un impact (baisse de similarité) direct sur les couples en liaison avec le couple de classe **(Person, Car)**. Cette baisse est prépondérante au poids associé à la catégorie classe dans le contexte d'une autre catégorie.

Pour remédier à cela, le calcul de la similarité entre deux entités se restreint aux catégories d'entités en commun. Le calcul de la similarité par normalisation de poids se fait suivant la formule 4.6. Ainsi, pour une catégorie X, la mesure de similarité par uniformisation de poids est comme suit :

$$Sim_X^+(x, x') = \frac{Sim_X(x, x')}{\sum_{\mathcal{F} \in \mathcal{N}^+(x,x')} \pi \mathcal{F}} \tag{4.6}$$

$\mathcal{N}^+(x, x')$ est l'ensemble des relations $\mathcal{F}$ où les $\mathcal{F}(x) \cup \mathcal{F}(x') \neq \emptyset$. La similarité du couple est normalisée par sa division sur la somme des poids associés aux catégories d'entités communes au couple.

4.3.4 Processus de calcul de similarités

Le calcul de la similarité globale se base sur un système d'équations quasi-linéaire dérivé du modèle de similarité global. La résolution de ce système se fait par un processus itératif capable de palier au problème de circularité. Il fournit à la fin de calcul les similarités maximales des couples d'entités.

4.3.4.1 Traitement de la circularité

Le calcul de la similarité d'un couple d'entités en fonction des similarités des autres couples n'est pas toujours direct. En effet, il induit dans certains cas une récursivité. Cette dernière émane de la définition circulaire des entités elles mêmes. Dans l'exemple de la figure FIG. 4.4, la similarité du couple d'individus *(Sean, Luc)* dépend de la similarité des deux propriétés *(MariedTo, MariedTo)* qui lient respectivement l'individu *Sean* à l'individu *Faith* et l'individu *Luc* à celui *Karine*, la similarité du couple de propriétés dépend à son tour de la similarité du couple *(Faith, Karine)* qui, à son tour dépend de la similarité du couple de départ *(Saen, Faith)* via le couple de propriétés *(MariedTo, MariedTo)*. Par conséquent, le calcul direct de la similarité n'est pas possible car on se retrouve dans un cas circulaire où la similarité du couple de départ dépend d'elle même. Il faudrait alors, trouver un moyen pour traiter les dépendances mutuelles.

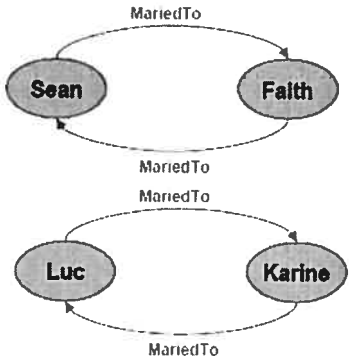

FIG. 4.4 – Exemple du cas de la circularité

Pour faire face à cette circularité, le calcul de la similarité est traduit sous forme d'un système d'équations dont la résolution se fait d'une manière itérative.

4.3.4.2 Système d'équations à point fixe

La généralisation du principe de calcul déduit de la formule 4.2 sur les différentes catégories fait sortir un système d'équations quasiment linéaire. Les variables de ce système sont les similarités des couples d'entités et les cœfficients représentent les poids associés aux catégories d'entités. La résolution du système d'équations est un problème d'optimisation dont la solution, après convergence, est le vecteur de valeurs de similarité calculées. La résolution du système se fait comme suit.

A l'itération 0 : les similarités lexicales ou terminologiques des URIs de couples d'entités par catégories sont calculées. En effet, le calcul de ces similarités se fait une seule fois tout au début du processus.

A l'itération 1 : le processus commence par calculer la similarité de couples d'entités par catégorie. Les similarités des catégories intervenant dans le calcul d'un couple lors de cette itération sont issues de l'itération 0 et agrégées suivant la formule 4.2

A l'itération i : le processus procède de la même manière en se basant sur les similarités calculées à l'itération $i-1$.

Par conséquent, à chaque itération la similarité des couples augmente par la contribution de la similarité des couples d'entités voisines. Ce mécanisme est applicable pour tous les couples. A titre illustratif,

la similarité des couples de "types de donnée" vont contribuer à la similarité des couples de propriété de nature "type de donnée" qui à leurs tours, leurs similarités vont participer à la similarité des couples des classes ayant ces propriétés. Ceci s'effectue par un effet de propagation de similarité à travers les couples d'entités. On remarque que dans le même exemple que les types de données, se trouvant au $2^{ème}$ niveau du OL-Graph, contribuent à la similarité des couples de classes via les propriétés exprimées sur ces classes. En outre, la nature "itérative" du processus permet de ramener la similarité des couples éloignés à ceux en liaison jusqu'à convergence du système. Cette dernière est assurée puisque la fonction de la similarité est positive, monotone et bornée par 1. En effet, après un certain nombre d'itérations le gain en similarité pour tous les couples sera négligeable, au quel cas le système sera considéré comme convergent. Le gain minimal de similarité entre deux itérations successives sera déterminé par un seuil ϵ fixe. En d'autres termes, le processus de calcul converge lorsque le gain de similarité pour tous les couples d'entités est inférieur à ce seuil. Dans le cas des entités parfaitement similaires le processus converge avec des valeurs égales à 1.

4.3.5 Production d'appariement

La production d'appariement constitue l'étape finale du processus d'alignement. Un appariement peut être :

- Complet/total : toutes les entités de la première ontologie seront appariées à celles de la deuxième.
- Injectif : toute entité de la première ontologie est appariée à une seule de la deuxième ontologie et vice-versa.

La production d'appariement se fait suivant un schéma d'appariement qui indique la façon d'apparier les entités de la première ontologie aux entités de la deuxième. Il existe différents schémas d'alignement qu'on peut produire comme résultat :

- $(1,1)$ une entité ne peut être alignée qu'à une seule.
- $(1,n)$ une entité peut être alignée à une ou plusieurs
- (n,n) la forme la plus générale.

4.3.5.1 Appariement à base de seuil

L'opération d'appariement peut être laissée au spécialiste qui déciderait quels couples d'entités à retenir en fonction de leurs valeurs de similarité.

Lorsque la complétude d'alignement n'est pas exigée, un filtrage à base d'un seuil permettra de retenir les entités les plus similaires. Ainsi, plusieurs méthodes à base de seuil sont présentées dans la literature [Ehrig et Sure, 2004]. Dans notre contexte, nous a adopté la méthode *Hard threshold*. La méthode retient toutes les correspondances dont la similarité est supérieure à un seuil n fixé par l'utilisateur.

4.3.5.2 Optimisation des résultats d'appariement

Lorsque l'injectivité d'appariement n'est pas exigée, la qualité d'alignement peut être maximisée sur la similarité totale des couples alignées. Par conséquent, l'algorithme d'alignement doit optimiser le critère global au lieu de maximiser la similarité locale des couples d'entités. Dans un premier lieu, nous avons opté pour un algorithme glouton dans le choix de couples d'entités les plus pertinentes. En effet, à chaque étape, l'algorithme choisit un couple d'entité ayant la plus grande similarité et ôte les deux entités du couple de la table des similarités. L'algorithme s'arrête lorsqu'il n'y plus de couples dont la similarité est supérieure au seuil fixé.
Quoi que la stratégie gloutonne ne soit pas toujours optimale, elle s'avère plus avantageuse dans beaucoup de situations. Cependant, le choix d'une autre méthode, pour mémoire la méthode hongroise, [Valtchev, 1999] peut donner de résultats plus intéressants du moment qu'elle est plus adaptée à ce genre de problème dit problème d'affectation.

4.3.6 Exemple de calcul illustratif

Afin d'avoir une idée sur le processus de calcul de similarité, nous présentons le cas de calcul de similarité entre les couples d'entités. Les entités sont tirées de deux ontologies représentées respectivement par les deux OL-Graphes (voir FIG.4.5 et FIG. 4.6). Les similarités résultantes sont données sous forme de tables : chaque ligne représente un couple d'entité et une colonne i, donne les similarités calculées à l'itération i. Du moment que le calcul suit la même logique pour toutes les catégories, nous nous sommes limité à expliciter le détail de calcul pour un couple de classes. La généralisation de l'approche de calcul sur les autres couples d'entités s'effectue de la même manière.

Paramétres de calcul

Il faut préciser que pour des raisons pédagogiques, nous avons pris les considérations suivantes :

1. Les deux ontologies choisies sont de petite taille.
2. Le calcul de similarité terminologique est fait par l'API de WordNet.
3. Les valeurs des poids[2] associés aux catégories d'entités sont des valeurs typiques détérminés par l'usager (la table TAB. 4.8 des poids fournis par l'usager) ou automatiquement par des procédés algorithmiques.
4. Le seuil d'arrêt est fixé à la valeur 0.01.

[2]Les valeurs des poids ont un impact sur les résultats du calcul des similarités. Dans cet exemple, aucune considération quand au choix optimal de ces valeurs n'a été prise.

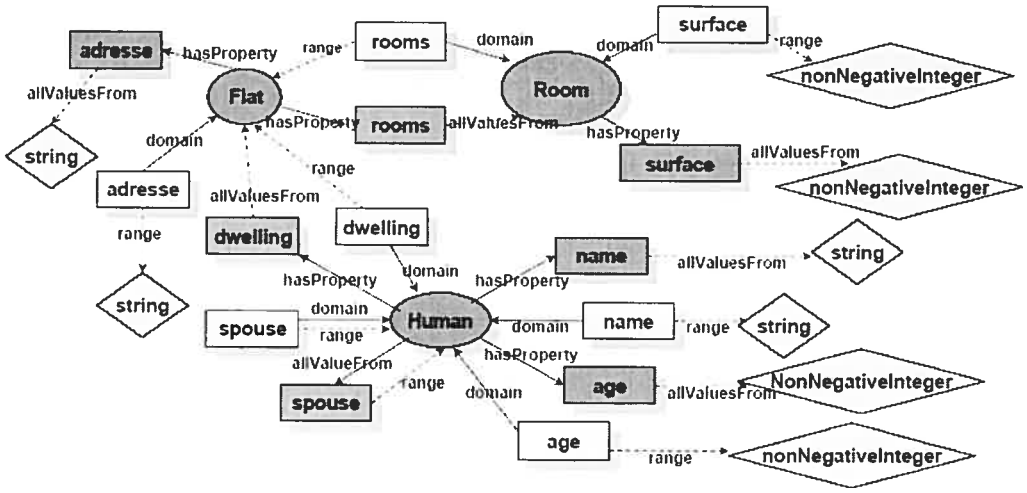

FIG. 4.5 – OL-Graph : première ontologie

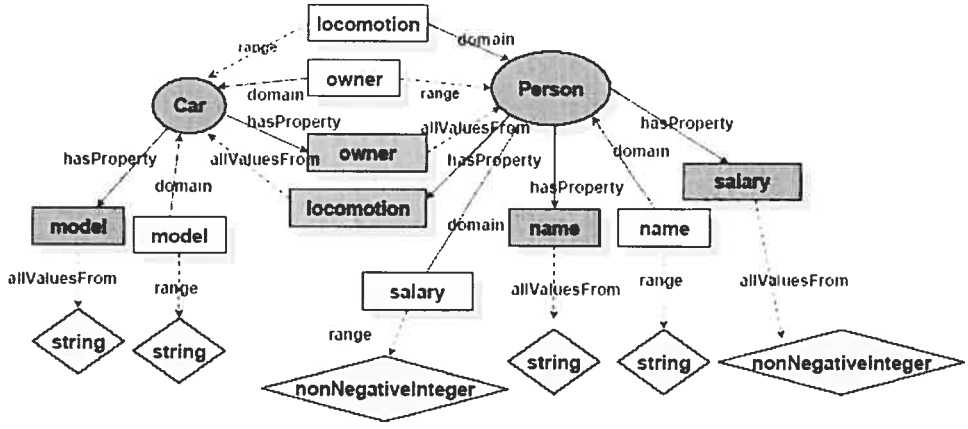

FIG. 4.6 – OL-Graph : deuxième ontologie

POIDS	VALEUR
CLASSE	
classLexicalWeight	**0.50**
classObjectPropertyWeight	**0.25**
classDatatypePropertyWeight	**0.25**
RELATION DE NATURE "OBJET"	
objectRelationLexicalWeight	0.40
objectRelationDomainWeight	0.30
objectRelationRangeWeight	0.30
RELATION DE NATURE "TYPE DE DONNÉE"	
datatypeRelationLexicalWeight	0.40
datatypeRelationDomainWeight	0.30
datatypeRelationRangeWeight	0.30
PROPRIÉTÉ DE NATURE "OBJET"	
objectPropertyLexicalWeight	0.20
objectPropertyClassWeight	0.40
objectPropertyRelationWeight	0.25
objectPropertyCardinalityWeight	0.15
PROPRIÉTÉ DE NATURE "TYPE DE DONNÉE"	
datatypePropertyLexicalWeight	0.20
datatypePropertyDatatypeWeight	0.40
datatypePropertyRelationWeight	0.25
datatypePropertyCardinalityWeight	0.15

TAB. 4.8 – Table des valeurs des poids

CLASSE	PROP. "OBJECT"	PROP. "TYPE DE DONNÉE"
Human	spouse dwelling	name age
Car	owner	model

TAB. 4.9 – Table des propriétés du couple de classes (Human, Car)

COUPLE	SIMILARITÉ LEX.	ITÉR. 1	ITÉR. 2	ITÉR. 3	ITÉR. 4
(Room, Car)	0.7000	0.5515	0.5515	0.5515	0.5515
(Room, Person)	0.2909	0.2422	0.2422	0.2422	0.2422
(Human, Car)	**0.2500**	0.3093	**0.3099**	0.3122	0.3122
(Human, Person)	0.7058	0.6224	0.6381	0.6412	0.6412
(Flat, Car)	0.6352	0.5891	0.5996	0.6044	0.6044
(Flat, Person)	0.2823	0.3650	0.3794	0.3829	0.3829

TAB. 4.10 – Table des similarités calculées entre les couples de classes

Résultats de calcul

Les similarités sont calculées pour toutes les catégories d'entités à savoir les classes, les relations, les propriétés de nature "objet" et de "type de donnée". Nous nous sommes restreint à présenter les similarités des couples de classes données par la table TAB. 4.10, les similarités des couples de propriétés de nature "objet" présentées dans la table TAB. 4.12 et les similarités des couples de propriétés de nature "type de donnée" représentées dans la table TAB. 4.11. Les similarités des autres catégories sont données en annexe par les tables TAB. A.5, TAB. A.4 et TAB. 4.9.

A titre illustratif, nous présentons le calcul de la similarité pour le couple de classes **(Human, Car)** à l'itération 2. Les deux classes ont les propriétés présentées dans la table TAB. 4.9.

4.3.6.1 Calcul de similarité pour le couple (Human, Car)

Le calcul de la similarité pour le couple **(Human, Car)**, selon la formule 4.7 et après le remplaçement des cœfficients par leurs valeurs, a donné les résultats présentés dans la table TAB. 4.8). Les catégories

COUPLE	SIMILARITÉ LEX.	ITÉR. 1	ITÉR. 2	ITÉR. 3	ITÉR. 4
(age, salary)	0.2222	0.7926	0.7926	0.7926	0.7926
(age, name)	0.2857	0.7824	0.7824	0.7824	0.7824
(age, model)	0.2500	**0.7728**	0.7728	0.7728	0.7728
(name, salary)	0.2000	0.7595	0.7595	0.7595	0.7595
(name, name)	1.0000	1.0000	1.0000	1.0000	1.0000
(name, model)	0.3817	**0.8351**	0.8351	0.8351	0.8351
(address, salary)	0.1538	0.7472	0.7472	0.7472	0.7472
(address, name)	0.6091	0.8958	0.8958	0.8958	0.8958
(address, model)	0.1667	0.7778	0.7778	0.7778	0.7778
(surface, salary)	0.1538	0.7744	0.7744	0.7744	0.7744
(surface, name)	0.2000	0.7595	0.7595	0.7595	0.7595
(surface, model)	0.3750	0.8062	0.8062	0.8062	0.8062

TAB. 4.11 – Table des similarités des couples de propriétés de nature "type de donnée"

COUPLE	SIMILARITÉ LEX.	ITÉR. 1	ITÉR. 2	ITÉR. 3	ITÉR. 4
(spouse, owner)	0.6250	**0.6444**	0.6626	0.6651	0.6651
(spouse, locomotion)	0.1250	0.3692	0.3814	0.3835	0.3835
(dwelling, owner)	0.1538	**0.3823**	0.3985	0.4011	0.4011
(dwelling, locomotion)	0.1111	0.4888	0.5141	0.5179	0.5179
(rooms, owner)	0.2000	0.3504	0.3693	0.3700	0.3700
(rooms, locomotion)	0.2667	0.5052	0.5194	0.5204	0.5204

TAB. 4.12 – Table des similarités des couples de propriétés de nature "objet"

	SIMILARITÉ
	owner
spouse	**0.6444**
dwelling	0.3823

TAB. 4.13 – Table des similarités des couples de propriétés "objet" du couple (Human, Car)

intervenant dans le calcul de la similarité du couple sont les propriétés de nature "objet" et les propriétés de nature "type de donnée".

$$Sim_{(Human,Car)} = 0.5 \times simL("Human","Car") \tag{4.7}$$

$$+ \; 0.25 \times Sim_P(\{name, age\}, \{model\}) \tag{4.8}$$

$$+ \; 0.25 \times Sim_P(\{spouse, dwelling\}, \{owner\}) \tag{4.9}$$

La valeur de la similarité lexicale $simL = 0.25$ des deux URIs "Human" et "Car" est calculée au début du processus et reste invariable. Le calcul itératif s'intéresse alors au calcul des similarités pour les deux catégories de nature "objet" et de nature "type de donnée". La formule 4.10 donne la formule de calcul de similarité pour le premier type de propriété.

$$Sim_P(Human, Car) = \frac{MSim((spouse, owner), (dwelling, owner)}{max(2, 1)} \tag{4.10}$$

$$= \frac{0.6444}{2} = 0.3222 \tag{4.11}$$

- La similarité $MSim(\{spouse, dwelling\}, \{owner\})$ représente la similarité des deux ensembles de propriétés de nature "objet". La similarité est calculée par un algorithme glouton. La table TAB. 4.13 donne le produit des deux ensembles avec les similarités correspondantes aux couples des propriétés. Les similarités de ces couples sont issues de l'itération 1 (colonne 1 de la table TAB. 4.12).

L'algorithme retient le couple *(spouse,owner)* ayant la plus grande similarité (0.6444), ensuite il enlève la ligne et la colonne (1,1) de la matrice. Le nombre d'itérations de ce processus est en nombre de la plus petite taille des deux ensembles. La similarité obtenue sera normalisée en la divisant sur le $max(|spouse, dwelling|, |owner|)$ qui est égale à 2.

- La similarité $Sim_P(\{name, age\}, \{model\})$ représente la similarité des deux ensembles de propriété de nature "type de donnée" du couple **(Human, Car)**. Le calcul est similaire à celui des propriétés de nature "objet". Le calcul est explicité par la formule 4.12.

$$Sim_P(\{name, age\}, \{model\}) = \frac{MSim((name, model), (age, model))}{max(2, 1)} \quad (4.12)$$

$$= 0.4175 \quad (4.13)$$

En remplaçant les termes des deux équations 4.10 et 4.12 dans la formule 4.7, on obtient la formule 4.14. La similarité obtenue est égale à celle calculée par l'algorithme (voir table TAB.4.10).

$$Sim_{(Human.Car)} = 0.5 \times 0.2500 + 0.25 \times 0.4175 + 0.25 \times 0.3222 \quad (4.14)$$

$$= 0.3099 \quad (4.15)$$

Appariement final produit

L'alignement produit de l'exemple 4.3.6 est donnée sous forme d'un fichier XML. Des détails sur les spécifications de ce format sont données en annexe.

```
<?xml version='1.0' encoding='utf-8' standalone='no'?> <rdf:RDF
xmlns='http://knowledgeweb.semanticweb.org/heterogeneity/alignment'
         xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:xsd='http://www.w3.org/2001/XMLSchema#'>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type></type>
  <uri1>file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl</uri1>
  <uri2>file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl</uri2>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#Flat'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#Person'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8333333333333334</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#address'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#name'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8831168831168831</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#age'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#salary'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8703703703703703</measure>
```

```
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#dwelling'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#locomotion'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.9111111111111111</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#spouse'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#locomotion'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.9125</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#Room'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#Person'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8666666666666667</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#Human'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#Person'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8636363636363636</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#rooms'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#locomotion'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.9266666666666666</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#surface'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#name'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8831168831168831</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-1.owl#name'/>
      <entity2 rdf:resource='file:/C:/Documents%20and%20Settings/ontoTest/my-onto-2.owl#name'/>
```

```
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
  </map>
</Alignment> </rdf:RDF>
```

4.4 Conclusion

Dans le précédent chapitre, nous avons présenté les fondements de l'approche d'alignement d'ontologies OWL-Lite. En effet, nous avons illustré le processus de construction d'OL-Graph ; une représentation qui reflète les entités et les relations d'une ontologie OWL-Lite. Ensuite, nous avons présenté le modèle de calcul de similarité entre les entités. Ainsi, nous avons expliqué les niveaux de calcul de similarité, et la dérivation du système d'équations dont la résolution permet de résoudre le problème de circularité des couples interdépendants. Finalement, nous avons élucidé la stratégie d'appariement adoptée et les moyens pour améliorer les résultats obtenus.

En somme, le modèle de similarité a fait preuve d'extensibilité. En effet, le modèle étant basé sur la structure d'OL-Graph peut être enrichi davantage. Il peut peut être étendu facilement pour inclure d'autres éléments de description d'entités. Le calcul de similarité par catégorie d'entités ajoute de la lisibilité au modèle et surtout la possibilité d'automatisation de l'opération d'alignement par la dérivation du système d'équations dont la résolution surmonte efficacement le problème de la circularité.

Dans le chapitre suivant, nous présentons les résultats des tests conduits pour valider la méthode et analyser la qualité des alignements produits. Ainsi, une base de tests fournis lors de la préparation de la compétition EON, met à l'épreuve notre approche et révèle les aspects qu'il faudrait ajuster pour aboutir à des résultats meilleurs.

Chapitre 5

Réalisation et expérimentation

Ce chapitre est organisé en deux parties. Dans la première partie, nous abordons les spécifications liées à l'implémentation de la structure d'OL-Graph et le modèle de calcul de similarité. Ainsi, nous présentons l'environnement de réalisation, les outils et les APIs utilisées. Afin de montrer nos réalisations, nous présentons les fonctionalités du système OLA sous forme d'un scénario d'exécution. Dans la deuxième partie, nous expliquons les tests conduits dans le cadre de notre participation à la compétition EON sur l'alignement d'ontologies [EON, 2004]. Finalement, nous commentons l'analyse des résultats obtenus et nous suggérons certaines améliorations à apporter.

5.1 Environnement de réalisation

L'environnement de développement est constitué d'outils suivants :

- Le langage de développement JAVA,
- L'environnement de développement *Eclipse*,
- L'outil de vérification et d'édition de fichiers XML *XMLSpy*

5.2 Réalisation d'OL-Graph

Pour implémenter l'OL-Graph, nous avons opté pour une modélisation de graphe par *arcs adjacents*. En effet, chaque nœud du graphe représente une entité de l'ontologie (classe, propriété, relation ..etc). Chaque nœud lui est associé les listes d'arcs adjacents rangées par catégorie.
L'implémentation de classes d'OL-Graph est très déclarative. Ainsi, chaque artefact (nœud ou arc) a été implémenté par une classe JAVA portant le même nom de l'artéfact sur l'OL-Graph concaténé au terme "Node" ou "Edge" pour désigner un nœud ou un arc respectivement. A titre d'exemple une entité de type classe dans le langage OWL-Lite est implémentée par une classe JAVA ayant le nom *ClassNode*. Une relation de spécialisation entre deux classes est implémentée par une classe JAVA nomée *ClassSpecialisationEdge*. L'implémentation d'OL-Graph est présentée en détail dans l'annexe.

5.3 Implémentation de l'algorithme d'alignement

L'algorithme d'alignement procède en quatre étapes :

1. Paramètrage,
2. Chargement des deux ontologies à aligner et construction d'OL-Graphs,
3. Calcul de similarités entre les couples d'entités,
4. Génération de l'alignement suivant le schéma spécifié.

5.3.1 Paramètrage

L'initialisation de paramètres consiste à spécifier les valeurs d'entrée de l'algorithme de calcul de la similarité. L'initialisation comprend :

- la spécification de différents poids associés à chaque catégorie d'entité,
- la spécification du seuil d'arrêt,

- la spécification du type d'algorithme pour le calcul de similarité terminologique/ lexicale,
- la spécification du schéma d'alignement à produire.

5.3.1.1 Initialisation des poids associés aux entités

L'initialisation des poids permet de spécifier pour chaque catégorie d'entité les poids des autres entités contribuant à sa similarité. A titre d'exemple : il faut spécifier combien une propriété de nature "type de donnée" contribuera dans le calcul de similarité d'un couple de classes. La table TAB. 5.1 donne les noms de paramètres et leurs designations. Les poids sont toujours normalisés. Autrement dit, leur somme devrait être égale à 1. Dans la version API du système OLA [Euzenat et al., 2004b] les paramètres sont lus à partir d'un fichier XML.

5.3.1.2 Choix du seuil d'arrêt

Le seuil d'arrêt représente une valeur réelle ϵ comprise entre $[0.1]$. Cette valeur devrait être si petite pour que les résultats finaux de l'alignement soient satisfaisants. Elle représente la différence de similarité calculée à l'étape n et l'étape $n-1$ du processus de calcul de similarité pour n'importe quel couple d'entités. Concrètement, le processus de calcul de similarité continue tant que pour au moins un couple d'entités, la similarité gagnée de l'étape précédente est supérieure au seuil ϵ. Il est clair que plus la valeur ϵ est petite, plus les valeurs finales de similarité calculées seront maximisées.

5.3.1.3 Choix de l'algorithme de calcul de similarité lexicale

L'approche d'alignement implémentée fournit la possibilité de choisir entre plusieurs algorithmes de calcul de similarité, celui à utiliser pour calculer la similarité de base (lexicale ou terminologique) entre les noms d'entités. Dans un premier temps, l'utilisateur peut choisir entre deux algorithmes :

- l'algorithme de calcul de similarité terminologique basée sur le *WordNet*. Ce dernier calcule la similarité terminologique en faisant recours à une utilisation du dictionnaire anglais.
- l'algorithme de calcul de similarité lexicale à base de la mesure de *SubStringDistance*.

La liste des algorithmes est extensible. En effet, les noms d'algorithmes sont spécifiés dans un fichier XML nommé *parameters.xml* extensible pour supporter d'autres méthodes.

5.3.1.4 Spécification du schéma d'alignement

La spécification du schéma d'alignement à produire consiste à choisir le type d'appariement entre les entités alignées. Du moment que l'alignement revient à choisir pour chaque entité la plus proche à une autre, notre approche prévoit un alignement du premier type.

NOM DU PARAMÈTRE	DESIGNATION
classObjectWeight	poids d'objet dans une classe
classSuperClassWeight	poids d'une super classe dans une classe
classDatatypePropertyWeight	poids de propriété de nature "type de donnée" dans une classe
classObjectPropertyWeight	poids de propriété de nature "object" dans une classe
objectPropertyClassWeight	poids de la classe dans d'une propriété de nature "objet"
objectPropertyObjectWeight	poids de l'objet dans une propriété de nature "objet"
datatypePropertyDatatypeWeight	poids du "type de donnée" dans une propriété de même nature
datatypePropertyDataValueWeight	poids d'une valeur de donnée dans une propriété de nature "type de donnée"
objectObjectPropertyInstanceWeight	poids d'une instance de propriété de nature "objet" dans un objet
objectDatatypePropertyInstanceWeight	poids d'une instance de propriété de nature "type de donnée" dans un objet
objectPropertyInstanceObjectWeight	poids de l'objet dans une instance de propriété de nature "objet"
datatypePropertyInstanceDataValueWeight	poids de la valeur de donnée dans une instance de propriété de nature "type de donnée"

TAB. 5.1 – Table des poids associés aux catégories d'entités

5.3.2 Chargement d'ontologies

Le chargement d'ontologies à aligner se fait une par une. Chaque ontologie, spécifiée par son URI, (URI pourrait pointer sur une ontologie publiée sous le Web ou un fichier local) sera parsée et transformée en un OL-Graph.

5.3.3 Calcul de la similarité

Le calcul de similarité s'effectue après la formulation du modèle de similarité sous forme d'un système d'équations quasi-linéaire. La première étape consiste à calculer la similarité lexicale ou terminologique entre les URIs d'entités ou tout autre élément de l'entité, et une deuxième étape se fait sous forme d'un processus itératif visant à résoudre le système d'équations. A la fin du processus, les similarités calculés représentent les similarités maximales atteintes.

5.3.3.1 Calcul de similarité terminologique/lexicale

Le calcul est effectué utilisant le nom de l'algorithme spécifié par l'utilisateur lors de l'étape de paramètrage. Le calcul consiste à calculer la similarité terminologique ou lexicale entre les littérales des entités par catégorie d'entités d'OL-Graph (type de données, propriétés, objets...). Les valeurs de la similarité terminologique seront rangées dans des matrices par catégorie d'entité de taille $[M, L]$, où M est la taille de la liste d'entités de la catégorie en question de la première ontologie et L est la taille de la liste d'entités de la deuxième ontologie de même catégorie. Le calcul de similarité lexicale se fait par les deux méthodes *Leveistein* et *SubString*. Le calcul de la similarité terminologique se fait à l'aide de la librairie JWNL.
Du fait que les noms d'entités dans les ontologies pourraient avoir de nomenclatures différentes incluant des abréviations, de noms concaténés..etc, il fallait utiliser les interfaces de la librairie JWNL combinées à d'autres techniques d'analyse lexicale pour rapprocher plus précisément le calcul de la similarité terminologique des URIs d'entités.

5.3.3.2 Processus itératif de calcul de similarité

Le calcul de la similarité se fait par la résolution du système d'équations. Les variables des équations sont remplacées par les couples d'entités à aligner. Ainsi, pour chaque catégorie d'entité, il y a trois matrices de taille $[L, M]$.

- une première contenant les similarités terminologiques calculées à l'étape 0,
- une deuxième matrice contenant les similarités calculées à l'itération $i-1$,
- une troisième matrice contenant les similarités calculées à l'étape courante i,

Les trois matrices sont de taille $[L, M]$ ou L le nombre d'entités de la première ontologie et M le nombre d'entités du même catégorie de la deuxième ontologie. A chaque itération, le calcul de similarité se fait pour tous les couples par catégorie d'entité.
Pour résoudre le problème de dépendances de similarité d'un couple aux autres couples, les valeurs de similarité de ces derniers, calculées à l'étape précédente, seront injectées dans le calcul de l'étape courante.

5.3.4 Production du schéma d'alignement

Il s'agit dans cette étape de produire un appariement entre les entités à base de valeurs de similarité calculées. La production d'appariement peut être effectuée manuellement par un expert qui juge de la pertinence des couples d'entités alignées. L'opération au fond consiste à choisir le couple d'entité ayant la plus grande valeur de similarité. Cependant, ce processus est laborieux.
Pour aider à l'automaticité de l'opération, nous avons opté pour un appariement automatique. Dans un premier lieu, l'algorithme d'appariement implémenté procède par une stratégie gloutonne. Ainsi, pour le choix d'entités à apparier l'algorithme retient parmi les couples alignés ceux ayant la plus grande valeur de similarité. Les entités ainsi choisies sont ôtées de la liste.
La stratégie gloutonne a affiché de bons résultats, toutefois d'autres techniques à base de la programmation dynamique ou de techniques de coloration de graphe s'avèrent plus efficaces.

5.4 Présentation d'OLA

OLA [Euzenat et al., 2004b] (Ontology Light Alignment) est la concrétisation de travaux sur la visualisation et l'alignement d'ontologie OWL-Lite effectués en collaboration avec l'équipe INRIA. OLA est un outil d'alignement des ontologies OWL open source ouvert. Les modules d'OLA sont organisés comme suit :

1. Le module de parsing des ontologies OWL-Lite,
2. Le module de visualisation des ontologies OWL-Lite,
3. Le module d'alignement des ontologies OWL-Lite,
4. Le module de production manuelle d'alignements,
5. Le module de visualisation d'alignements,
6. Le module de comparaison d'alignements.

5.4.1 Librairie OWL-API

L'OWL-API [Bechhofer et al., 2003] est une première version de bibliothèque développée par le *Web ontology Working Group* offrant un ensemble d'interfaces pour le parsing, l'accès et la manipulation des ontologies OWL. L'objectif visé par le développement de cette API est de permettre l'échange d'ontologies entre les applications à travers le Web en exemptant l'utilisateur des détails liés à la sémantique, la syntaxe du langage OWL et la sérialisation d'objets. L'utilisation de cette API permet le parsing d'ontologies et l'extraction de différentes entités et relations contenues dans une ontologie OWL-Lite.

5.4.2 Librairie JWNL de WordNet

JWNL (Java WordNet Librairy) [Miller, 1995] est une API qui donne accès à un ensemble de fonctionalités de recherche dans les dictionnaires en anglais, de calcul de similarité entre les noms, et de comparaison morphologique de noms. La *JWNL* implémente les résultats de nombreux travaux de recherche conduits par le laboratoire des sciences cognitives de l'université Princeton. Les travaux ont donné naissance au projet WordNet. L'utilisation de la librairie, dans sa version 1.3, nous a servi pour calculer la similarité terminologique entre les noms d'entités. En effet, la librairie permet entre autre de renvoyer les listes de synonymes et hyponymes communs entre deux noms. Ces listes peuvent être exploitées pour calculer des mesures de similarité terminologique plus précises.

5.4.3 Outils d'édition et de validation d'XML

Les outils d'édition et de validation de fichiers XML sont nombreux. Deux outils ont été utilisés :

- *XML-Spy* : un plug-in intégré à Eclipse. Il permet la visualisation d'ontologies OWL sous forme d'une arborescence de balises XML.
- *Amaya* : un outil élaboré par W3C incluant une application collaborative pour l'édition, la validation et l'annotation des fichiers sous format XML et RDF.

Vu que les ontologies sont reconnues comme des fichiers XML, ces outils nous ont permet d'éditer et de produire des ontologies de tests pour des validations plus rapides.

5.4.4 Outils de visualisation d'ontologies

La visualisation d'ontologies est un moyen efficace pour effectuer des validations et des éditions d'ontologies. Toutefois, peu d'outils offrent la convivialité et la gestion du layout nécessaires pour de grand-ontologies. Pour combler notre besoin, nous avons fait usage de deux outils.

- ***ViSon*** : un outil de visualisation développé par l'équipe interne de l'université de Montréal[1].
- ***Protégé 2000*** : via le plug-in ontoViz intégré. Le plug-in permet la visualisation d'ontologies OWL suivant le modèle des Frames.

Il faut signaler que *ViSon* est plus adapté à nos besoins du fait qu'il permet une visualisation d'ontologies suivant le modèle OL-Graph. *ViSon* a été intégré à l'outil OLA dans la perspective d'élaborer un environnement d'édition visuel complet.

5.4.5 Visualisation d'OL-Graph par VisOn

Le module fournis une interface graphique FIG. 5.1 de visualisation d'ontologies suivant les artéfacts d'OL-Graph. Ainsi, il constitue un moyen efficace de validation de parsing d'ontologies OWL-Lite. Le module dans sa première version, constitue le noyau pour la conception d'un outil d'édition et de manipulation d'ontologies OWL-Lite. *ViSon* prend en entrée une ontologie OWL-Lite dont l'URI est spécifié par l'utilisateur. L'ontologie est ensuite parsée et transformée sous forme d'OL-Graph. L'interface permet un accès par nom d'objet à partir de la liste d'entités d'OL-Graph.
En outre, *ViSon* permet aussi une visualisation juxtaposée de deux ontologies lors d'une préparation pour le lancement d'une opération d'alignement. Ainsi, l'utilisateur pourrait charger et visualiser les deux ontologies en question (voir la figure FIG. 5.2).

Les améliorations à apporter sur l'outil concernent :

- la gestion de layout pour une visualisation plus efficace d'ontologies,
- l'accès aux objets graphiques,
- l'édition d'ontologies via des opérations d'ajout, de suppression et de modification d'objets graphiques.

5.4.6 Calcul de similarité : alignement

Le module d'alignement est une implémentation de notre approche d'alignement. Le module fournit les interfaces pour paramétrer le processus d'alignement, la spécification du type d'algorithme à exécuter, visualisation de l'alignement produit et la production d'un pré-alignement manuelle.

[1]L'outil a été développé dans le cadre d'un stage de fin d'étude.

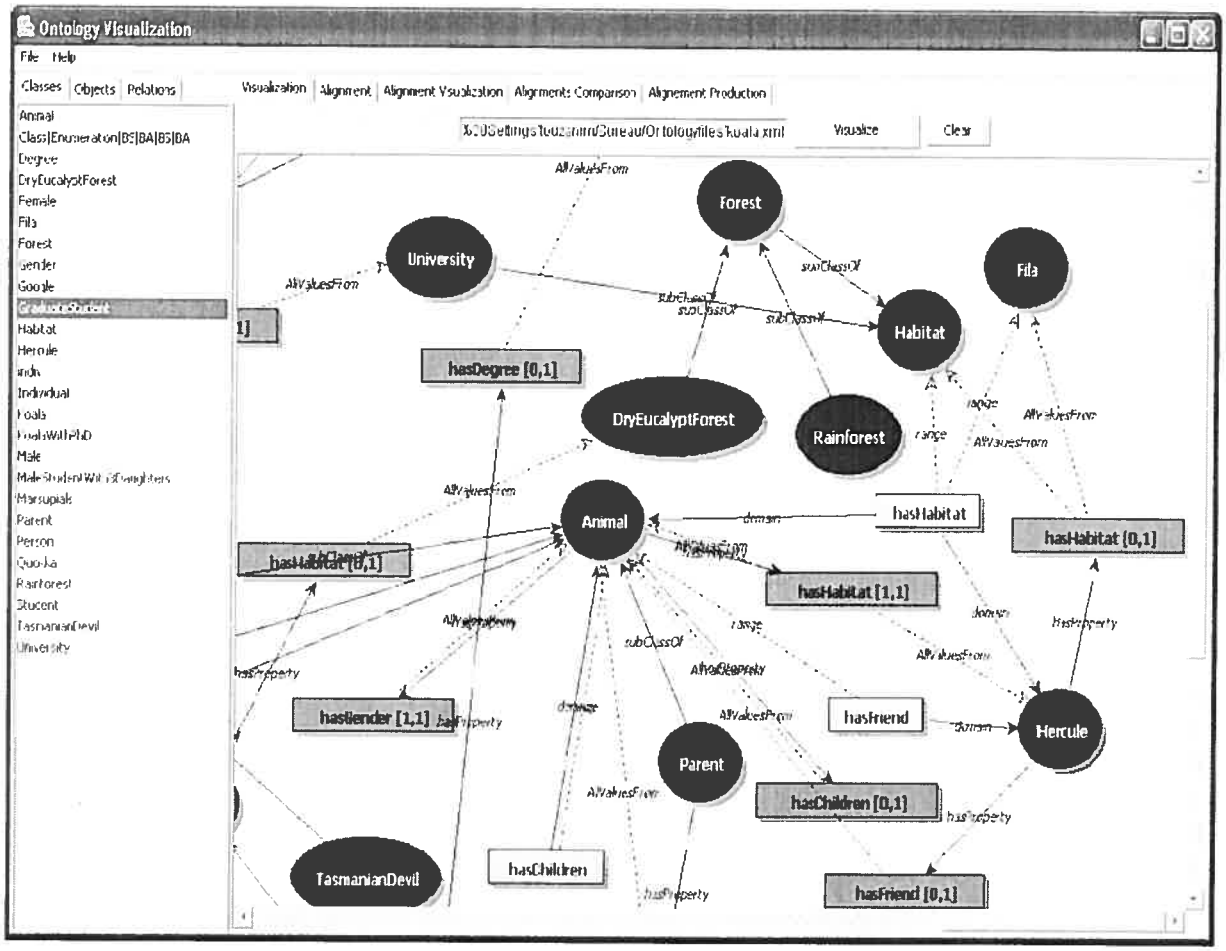

FIG. 5.1 – Interface de visualisation d'ontologie sous forme OL-Graph

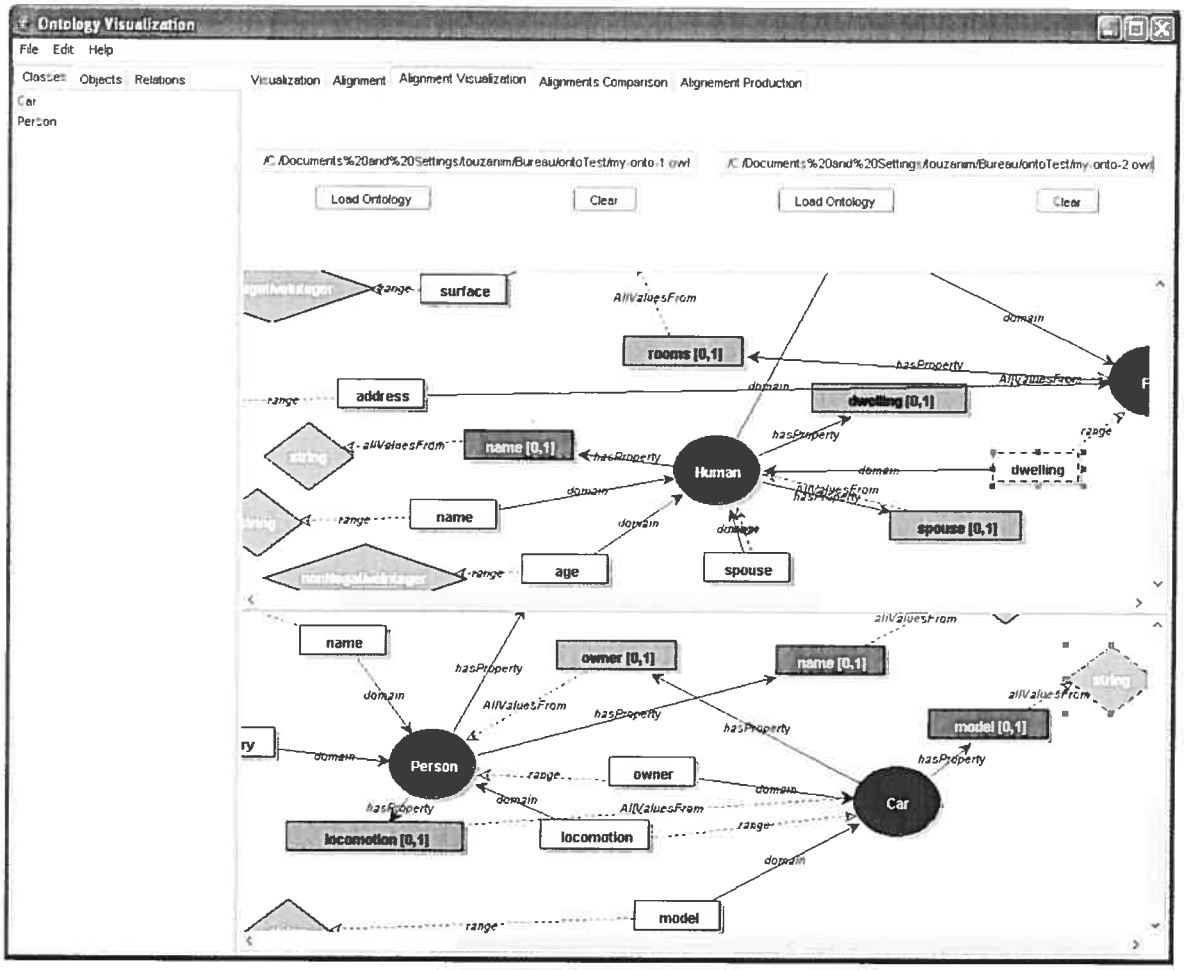

FIG. 5.2 – Interface de visualisation juxtaposée des ontologies

5.4.6.1 Spécification des poids et du seuil

Le choix de paramètres[2] se fait via l'interface FIG. 5.3. Les valeurs par défaut sont chargés à partir d'un fichier XML (voir l'annexe) ou spécifiés par l'utilisateur.

5.4.6.2 Spécification de la fonction de similarité lexicale

L'interface FIG. 5.4 permet de spécifier le type d'algorithme de calcul de similarité lexicale. La liste des fonctions est extensible : l'utilisateur à la possibilité d'implémenter d'autres algorithmes de calcul de similarité à base d'autres méthodes.

Deux méthodes de calcul de similarités respectivement terminologique et lexicale sont implémentées :

- la similarité terminologique utilisant la bibliothèque de *WordNWet*,
- la similarité lexicale utilisant l'algorithme *SubString*.

5.4.6.3 Spécification du type d'algorithme

OLA est une plate-forme ouverte qui peut intégrer plusieurs algorithmes d'alignement. L'interface FIG. 5.5 de spécification du type d'algorithme permet de choisir parmi la liste des algorithmes disponibles, celui à exécuter pour obtenir l'alignement. Dans notre cas, deux algorithmes sont implémentés utilisant le même modèle de calcul de similarité :

- le premier produit un alignement automatique à base de deux OL-Graphs.
- le deuxième, -une extension du premier- en plus des deux graphes, prend un alignement produit en avance. Les similarités des couples de cet alignement seront injectées lors du calcul de similarités des couples restants. Ainsi, cet algorithme permet de produire un alignement automatique, en se servant d'un autre alignement manuel -souvent partiel-, dont les valeurs des similarités des couples d'entités sont specifiées par l'utilisateur.

5.4.6.4 Spécification du schéma d'alignement

L'interface de spécification du schéma d'alignement FIG. 5.6 permet de déterminer le schéma d'alignement à produire. Les schémas d'alignement proposés sont $(1,1)$, $(1,n)$, (n,n).

[2]Dans une version amélioré d'*OLA*, les poids par défauts représentent la combinaison optimale qui donne le meilleur alignement. La spécification de ces poids serait le résultat des tests utilisant les techniques apprentissage machine.

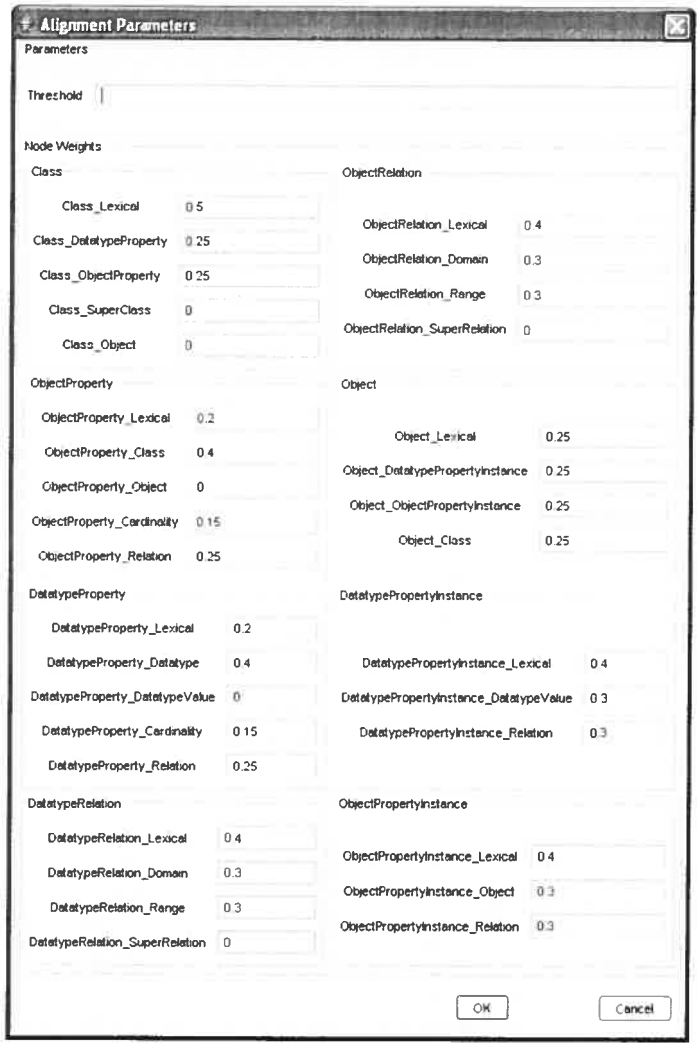

FIG. 5.3 – Interface de spécification des poids

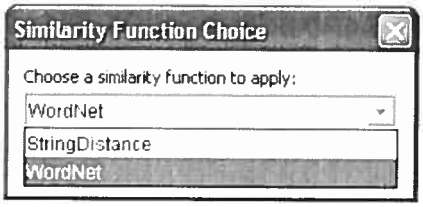

FIG. 5.4 – Interface de spécification de la fonction de similarité lexicale ou terminologique

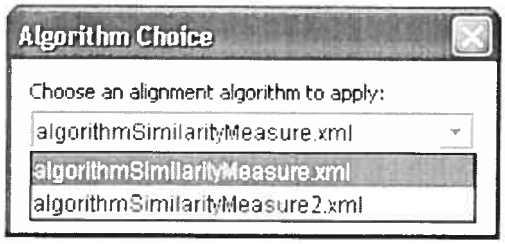

FIG. 5.5 – Interface du choix d'algorithme d'alignement

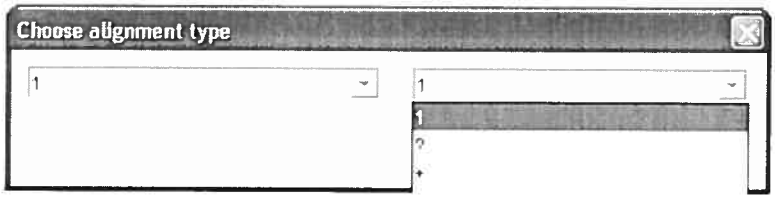

FIG. 5.6 – Interface du choix du schéma d'alignement

FIG. 5.7 – Interface d'affichage du résultat d'alignement

5.4.6.5 Affichage des résultats d'alignement

Suite à une opération d'alignement, le résultat est affiché sous forme d'une grille FIG. 5.7 dont les deux colonnes donnent les deux entités alignées et la troisième colonne la valeur de la similarité calculée par l'algorithme. Les couples d'entités alignés sont organisées en trois catégories :

1. les couples de classes,
2. les couples de relations objet et "type de donnée",
3. les instances.

5.4.6.6 Enregistrement d'alignement

L'alignement produit par l'algorithme est visualisé sous un format XML 5.8 (le format de ce fichier avec un exemple sont donnés en annexe) et peut être enregistré localement.

5.4.7 Production manuelle d'alignement

L'interface de production d'alignement FIG. 5.9 permet de produire des alignements manuellement. L'interface fournit les entités des ontologies parsées et chargées à l'avance. L'expert peut spécifier pour

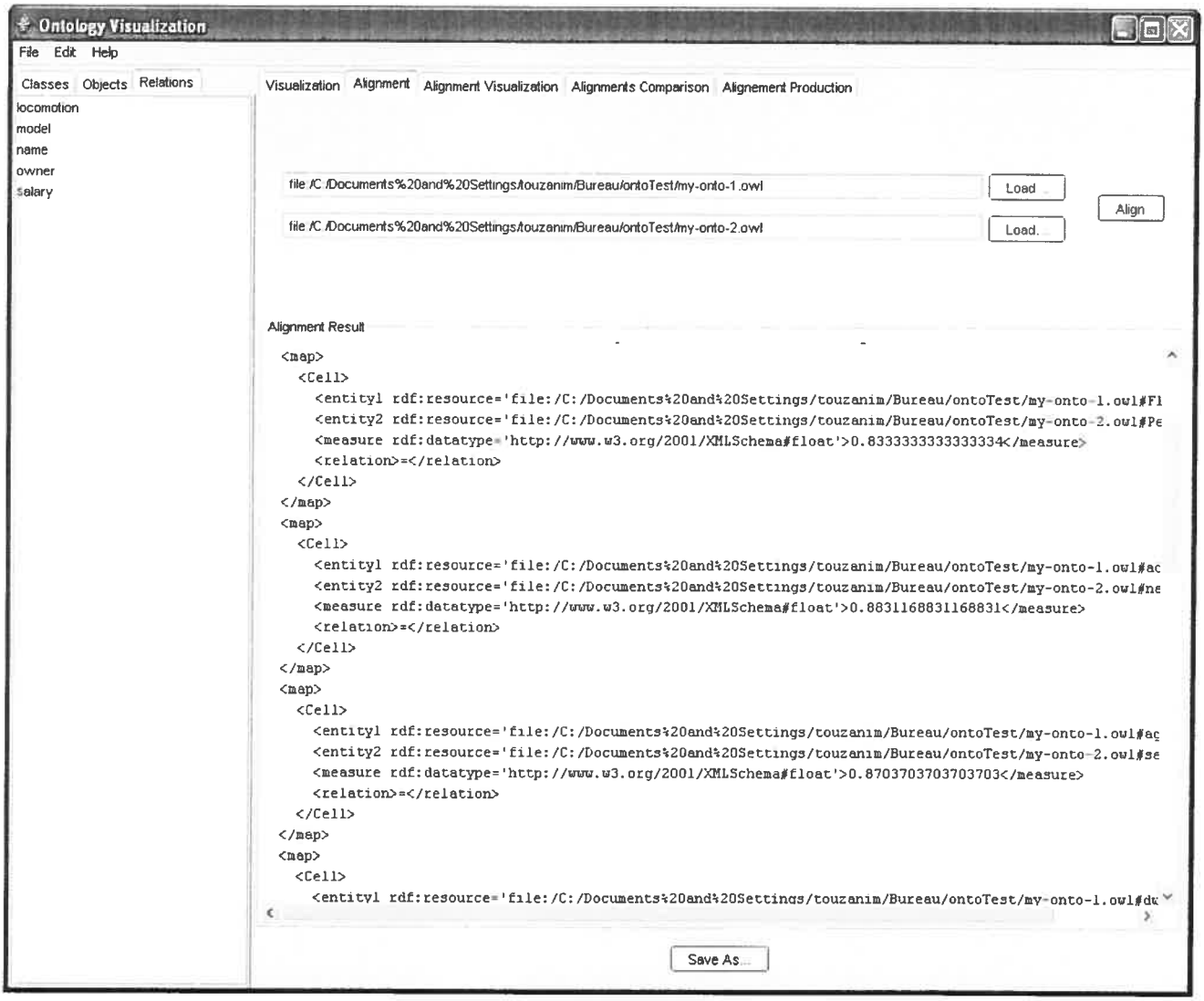

FIG. 5.8 – Interface de visualisation d'alignement sous forme d'un fichier XML

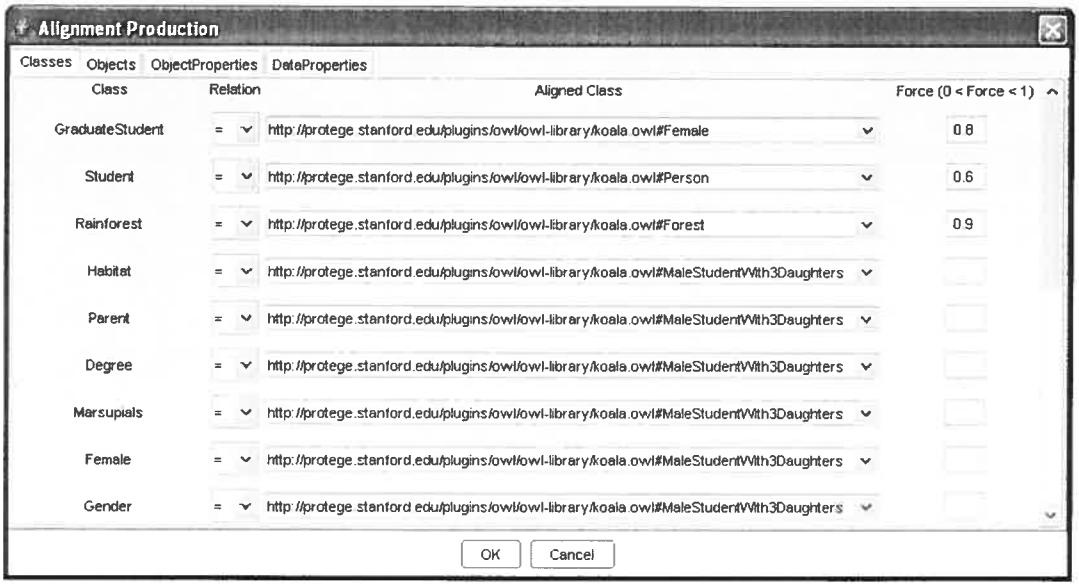

FIG. 5.9 – Interface de production d'alignement

des couples d'entités des valeurs de similarité selon le degré de ressemblance.
A la fin de l'opération, un alignement peut être produit et enregistré. Il pourrait servir comme un pré-alignement pour des opérations d'alignement ultérieures.

5.4.8 Comparaison d'alignements

OLA offre une interface FIG. 5.10 pour évaluer et comparer automatiquement deux alignements. L'interface permet de charger deux alignements sous forme d'un fichier XML et produit le résultat de la comparaison sous forme aussi d'un fichier XML[3].

5.5 Expérimentation et analyse des résultats

Tester les méthodes d'alignement constitue une étape difficile même délicate. Les travaux de recherche réalisés dans ce cadre sont encore dans un état embryonnaire. En effet, tester la pertinence d'un alignement n'est que l'opération de vérifier que telle entité est bel bien alignée avec celle la plus

[3]Les formats des deux fichiers sont décrits sur [Valtchev et al., 2004].

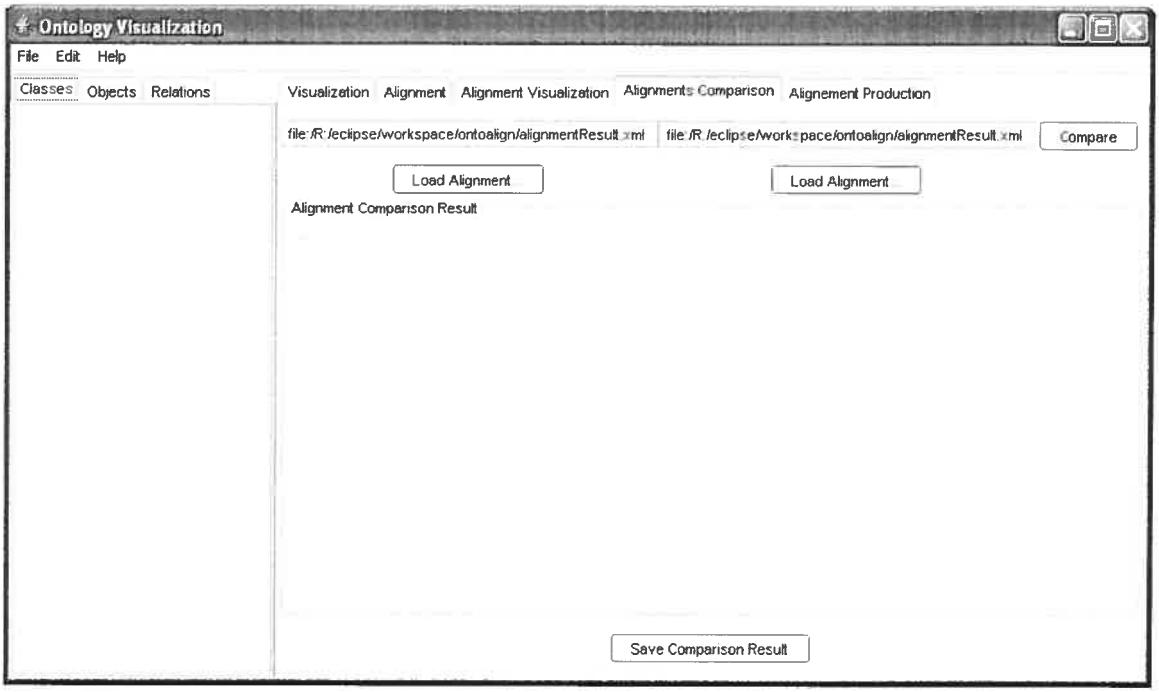

FIG. 5.10 – Interface de comparaison d'alignements

sémantiquement proche. Cette opération relève purement du domaine du spécialiste. Par conséquent, pour aider à l'automaticité de l'opération, il va falloir mettre à la disposition de développeurs des benchmarks (cas d'ontologies test) qui permettent de tester la validité et l'efficacité de leurs méthodes. Pour pousser les travaux dans ce sens, une première initiative était l'organisation de la compétition sur l'alignement d'ontologies organisée par EON [EON, 2004].
Notre participation à cette competition nous a offert l'occasion de valider notre approche et d'évaluer ces performances.

5.5.1 Tests préliminaires

Les tests préliminaires portent sur la vérification des résultats du parsing et la construction d'OL-Graph ainsi que la convergence de l'algorithme d'alignement. En effet, pour vérifier le parsing, nous avons adopté une stratégie incrémentale : En commençant par le parsing de petites ontologies, ayant un nombre petit d'entités et une description simple, à celles plus complexes. A chaque test, nous augmentons le nombre et la complexité des entités de l'ontologie et nous nous assurons des résultats obtenus. Afin d'accélérer ce processus, nous avons utilisé le module de visualisation des ontologies *ViSon* et l'outil XMLSpy.
L'éventualité d'avoir de la récursivité lors le calcul de similarité émane de la définition circulaire de certaines entités. Cette récursivité a été conquise par la dérivation des formules d'agrégation des similarités sous forme d'un système d'équations quasi-linéaires. Il faudrait cependant, vérifier la convergence du processus itératif qui en découle. Pour se faire, nous avons assujetti l'algorithme à des cas d'ontologies dont le nombre d'entités ayant une circularité très élevée.

5.5.2 Contexte général

Dans le cadre de notre participation à la compétition d'alignement organisée par l'EON, nous avons eu l'opportunité de tester la validité de résultats produits par notre approche d'alignement. La participation consiste à produire des alignements entre une ontologie OWL-Lite de base et un ensemble d'ontologies de tests décrits dans la table TAB.5.2. Le but par chaque cas de test est d'observer la performance de l'approche d'alignement face à une altération de l'ontologie.

5.5.2.1 Base de tests

La compétition EON a mis à la disposition de participants une base de tests [4] (table FIG. 5.2). L'ontologie de base est constituée par des références bibliographiques. L'ontologie représente une version limitée en nombre de classes et en propriétés comparativement avec une ontologie réelle. Pour étudier et analyser le comportement de leurs méthodes d'alignement, chaque cas de test met en exergue une caractéristique (un défis) de la deuxième ontologie à aligner avec l'ontologie de base. Le but par cette base de tests est de toucher à tous les aspects d'une ontologie OWL-Lite qui pourraient avoir un impact sur l'alignement produit en l'occurrence, la structure de l'ontologie, l'élément linguistique, la nomenclature de nommage d'entités, etc.

5.5.3 Choix de poids

Le choix des poids associés aux catégories d'entités a un impact direct sur les résultats d'alignement. Dans un premier, vu que le choix de la combinaison optimale des poids nécessite l'utilisation des techniques adaptées, le choix de poids pour les tests de compétition s'est basé sur des considérations simples :

- Attribuer des poids égaux pour tous les facteurs d'une catégorie d'entité. Par exemple donner une valeur de 0.2 pour tout facteur associé à la catégorie de type "classe",
- Attribuer un poids égale à 1 à un facteur et annuler les autres,
- Attribuer un poids propondérant (supérieur à 0.5) à la similarité lexicale.

5.6 Analyse des résultats

Afin d'évaluer la qualité d'alignements obtenus, trois critères sont retenus [EON, 2004]. Le but par la définition de ces critères est d'aider à l'évaluation des alignements et l'automatisation du processus de comparaison des alignement produits.

5.6.1 Critères d'évaluation

Ces critères sont compris dans $[0, 1]$ et sont définis comme suit :

- *fallout*

$$fallout = \frac{N_{Found} - N_{Correct}}{N_{Found}}; \tag{5.1}$$

[4] http ://co4.inrialpes.fr/align/Contest/

N. TEST	CARACTÉRISTIQUE DE L'ONTOLOGIE
101	même ontologie de base
102	le domaine d'intérêt est tout à fait différent de celui de l'ontologie de base.
103	les contraintes indisponibles sont remplacées par celles les plus générales disponibles.
104	les contraintes non disponibles ont été supprimées.
201	les libellés des entités sont remplacés par des libellés arbitraires.
202	les libellés sont remplacés par d'autres aléatoires. Les commentaires (rdfs :comment et dc :description) ont été supprimés.
204	des conventions d'appellation différentes (Uppercasing, soulignage, tiret, etc...) sont employées. Les commentaires ont été supprimés.
205	les libellés sont remplacés par des synonymes. Les commentaires ont été supprimés.
206	complétée est traduite à une autre langue que l'anglais (français dans le cas courant).
221	les assertions de sous-classe liées aux classes nommées sont supprimées.
222	la hiérarchie des classes est maintenue mais elle a été strictement réduite.
223	de nombreuses classes intermédiaires sont ajoutées à la hiérarchie.
224	les individus sont supprimés.
225	les restrictions locales exprimées par des propriétés ont été supprimées.
226	les types de données sont transformés de type "string".
228	les propriétés et les relations entre les objets ont été complètement supprimées.
230	quelques composants de classes sont étendus dans la structure de classe (date en année/mois/jours).
301	l'ontologie est réelle, plus simple et semblable à l'ontologie initiale de BibTeX[5].
302	l'ontologie est très semblable au précédente[6].
303	l'ontologie est réelle[7]
304	l'ontologie est réelle[8]. Sa hierarchie contient des classes qui sont des sous-classes de plusieurs autres classes. (fr.inrialpes.exmo.rdf.bib.owl)

TAB. 5.2 – Table des ontologies de test : base de tests

- *recall*

$$recall = \frac{N_{Correct}}{N_{Expected}}; \tag{5.2}$$

- *precision*

$$precision = \frac{N_{Correct}}{N_{Found}}; \tag{5.3}$$

ou le :

- N_{Found} : le nombre d'entités alignées par l'algorithme d'alignement,
- $N_{Correct}$: le nombre d'entités correctement alignées,
- $N_{Expected}$: le nombre d'entités alignées attendues (le nombre de couples alignées dans l'alignement de base).

5.6.2 Analyse et observations

Pour chaque test, l'alignement produit est comparé à celui de base d'où les résultats pour les paramètres dûment mentionnés sont calculés. Les résultats de notre approche sont résumés dans le graphe de niveaux FIG. 5.11 et la table TAB. 5.3. Dans cette section nous commentons les résultats de notre approche suivant les remarques tirées lors de la phase des tests. La table TAB. 5.4 donne en parallel les résultat des autres participants. Ainsi, nous mettons en lumière certains résultats.

A partir des données présentées dans la table TAB. 5.4 nous pouvons tirer les conclusions suivantes :

1. L'algorithme affiche de bons résultats lorsque les structures d'ontologies alignées sont semblables ou identiques (le cas d'ontologies $10X$ et $22X$). La précision d'alignement obtenue par notre algorithme est supérieure à 0.9. En effet, le calcul de la similarité repose considérablement sur les structures d'entités alignées. Ainsi, les entités ayant la même structure sont correctement alignées.
2. Les résultats de tests où la précision est faible, se justifient par le fait que notre algorithme aligne toutes les entités (instances, classes anonymes,...). Ceci fait que le nombre N_{Found} est élevé (exemple d'ontologies ayant un nombre d'instances élevé) et donc a un effet inverse sur la *précision.*
3. une partie des résultats mitigés s'explique par le fait que le calcul de similarité se limite aux entités de même catégorie. Ainsi, et suivant la manière de fonctionnement de l'algorithme, ceci affecte négativement l'alignement sur deux plans :
   - Certains couples d'entités sont exclus du processus de l'alignement. Par conséquent, le facteur *N_Correct* (nombre de de couples correctement aligné) est faible induisant des valeurs basses pour la *precision* et le *fallout* .

N. TEST	ASPECT	SIM. LEXI./TERM.	PRECISION	RECALL	FALLOUT
101	Id	SD	0.97	0.59	0.03
102	Irrelevant	SD	1.0	N/A	0.0
103	Language Généralisation	SD	0.901	0.550	0.099
104	Language Restriction	SD	0.912	0.557	0.088
201	No Names	SD	0.714	0.436	0.286
202	No Names, No Comments	SD	0.626	0.383	0.374
204	Naming Conventions	SD	0.901	0.550	0.099
205	Synonyms	WN	0.802	0.490	0.198
206	Foreign Names	WN	0.761	0.450	0.239
221	No Hierarchy	WN	1.0	0.611	0.0
222	Flattened Hierarchy	WN	0.901	0.550	0.099
223	Expanded Hierarchy	WN	0.967	0.590	0.033
224	No Instances	WN	1.0	0.968	0.0
225	No Restrictions	WN	0.967	0.590	0.033
228	No Properties	SD	1.04	0.375	0.000
230	Flattening Entities	WN	0.920	0.463	0.08
301	BibTeX/MIT	WN	0.607	0.493	0.393
302	BibTeX/UMBC	WN	0.500	0.226	0.500
303	Karlsruhe	WN	0.500	0.311	0.500
304	INRIA	WN	0.618	0.439	0.382

TAB. 5.3 – Table des résultats des tests

	KARLSRUHE2		MONTREAL		FUJITSU		STANFORD	
TEST	PREC.	REC.	PREC.	REC.	PREC.	REC.	PREC.	REC.
101	N/A	N/A	0.59	0.97	0.99	1.00	0.99	1.00
102	N/A	N/A	0.00	N/A	N/A	N/A	N/A	N/A
103	N/A	N/A	0.55	0.90	0.99	1.00	0.99	1.00
104	N/A	N/A	0.56	0.91	0.99	1.00	0.99	1.00
201	0.43	0.51	0.44	0.71	0.98	0.92	1.00	0.11
202	N/A	N/A	0.38	0.63	0.95	0.42	1.00	0.11
204	0.62	1.00	0.55	0.90	0.95	0.91	0.99	1.00
205	0.47	0.60	0.49	0.80	0.79	0.63	0.95	0.43
206	0.48	0.60	0.46	0.75	0.85	0.64	1.00	0.46
221	N/A	N/A	0.61	1.00	0.98	0.88	0.99	1.00
222	N/A	N/A	0.55	0.90	0.99	0.92	0.98	0.95
223	0.59	0.96	0.59	0.97	0.95	0.87	0.95	0.96
224	0.97	0.97	0.97	1.00	0.99	1.00	0.99	1.00
225	N/A	N/A	0.59	0.97	0.99	1.00	0.99	1.00
228	N/A	N/A	0.38	1.00	0.91	0.97	1.00	1.00
230	0.60	0.95	0.46	0.92	0.97	0.95	0.99	0.93
301	0.85	0.36	0.49	0.61	0.89	0.66	0.93	0.44
302	1.00	0.23	0.23	0.50	0.39	0.60	0.94	0.65
303	0.85	0.73	0.31	0.50	0.51	0.50	0.85	0.81
304	0.91	0.92	0.44	0.62	0.85	0.92	0.97	0.97

TAB. 5.4 – Table des résultats de partipants à la compétition EON

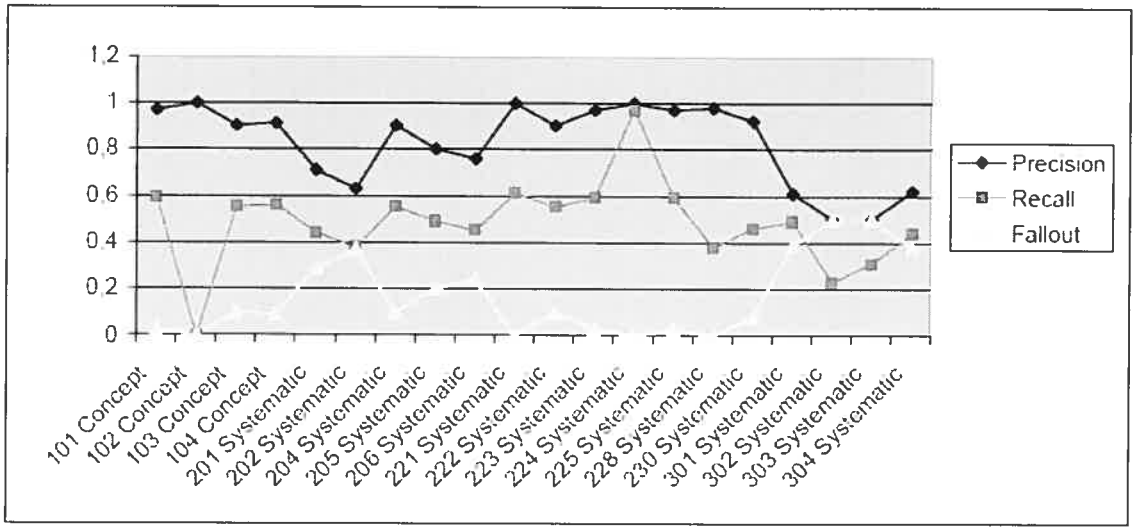

FIG. 5.11 – Graphe des résultats des tests

- Les couples exclus pourraient contribuer dans l'augmentation de la similarité des couples ascendants et par conséquent augmenter le nombre de couples lors de la production de l'alignement.

Pour commenter les résultats de la compétition, donnés dans la table TAB. 5.4, nous nous contetons des obsérvations suivantes[9] :

- La comparaison des résultats des participants s'est effectuée à base des deux mesures : *Recall* et *precision* qui sont faciles à comprendre mais par contre, sont inversement liées. En effet, l'augmentation de l'une résulte en une baisse de l'autre. Par conséquent, un algorithme peut avoir les mêmes résultats qu'un autre sans qu'ils soient clairement montrés par la table,
- Les meilleurs résultats sont affichés par les deux algorithmes de Stanford et de Fujitsu : Deux algorithmes très différents mais qui reposent considérablement sur l'analyse des libéllés des entités. Ce qui explique les bons résultats lorsque les libéllés sont préservés
- Notre approche et celle de Karlsruhe reposent sur plusieurs aspects entre autre l'intégration des individus dans le calcul de la similarité.

En général, les participants ont affichés de bons résultats pour les quatres premiers tests. Par contre, pour les tests systématiques, les deux algorithmes de Stanford et Fujitsu ont marqué des résultats remarquables. Toutefois, pour le cas des ontologies réelles, elles ont posé des difficultés pour tous les

[9]Certaines des obsérvations émises sont tirées du site de la compétition [EON, 2004].

participants. Reste que, par les jeux de tests, la compétiton a mis l'accent surtout sur l'acpect structurel et lexical d'ontologies malgré qu'une analyse plus rigoureuse devrait focaliser sur des tests où l'accent est mis sur l'aspect symantique.

5.6.3 Amélioration des résultats

Dans le but d'améliorer les résultats obtenus, nous avons dégagé trois volets susceptibles d'apporter une amélioration nette sur les résultats de l'approche d'alignement proposée :

1. **Calcul de similarité inter-catégorie** : le calcul de similarité inter-catégories va chercher la similarité entre les entités appartenant à des catégories différentes. En effet, les résultats des calculs manuels ont montré une augmentation aussi bien du nombre de couples correctement alignés que des valeurs des similarités. Ceci peut être expliqué dans le premier cas par l'ajout des autres couples dans la liste de couples candidats d'alignement et dans le deuxième cas par la combinaison de leurs similarité dans le calcul de similarité des couples ascendants.

   Le calcul de similarité inter-catégorie s'effectuera à priori entre les catégories suivantes.

   (a) Les propriétés de nature "objet" avec celles de nature "type de donnée" : ceci engendra une augmentation du nombre de couples correctement alignés et une amélioration de la similarité entre les couples de classes.

   (b) Les classes avec les types de donnée : ceci permettra une amélioration du nombre de couples correctement alignés et une augmentation des similarités des propriétés.

   (c) Les instances avec les valeurs : ceci aura un impact sur le nombre de couples correctement alignés et sur l'augmentation de la similarité des individus et par conséquent celles des classes.

   Par un effet de propagation des similarité, la similarité des autres couples d'entités augmenteront à leurs tours (voir 4.3.4 chapitre 2).

2. **Meilleur choix de poids** : il est clair qu'une meilleure combinaison de poids associés au catégories d'entités est un facteur déterminant pour ajuster le comportement de l'algorithme. En effet, l'utilisation des méthodes et des stratégies plus intelligentes pour déterminer des poids optimaux devrait avoir un impact positif sur la pertinence des alignements obtenus.
3. **Calcul plus fin de la similarité terminologique** : Par un calcul plus granulé de la similarité entre les URIs des entités, le gain en similarité terminologique contribuera significativement dans l'amélioration des similarités finales. Ceci s'explique par le fait que la similarité terminologique est un facteur clé dans le modèle de calcul de similarité global.

5.7 Conclusion

Ce chapitre a abordé les questions liées à l'implémentation de notre approche de similarité. En effet, l'implémentaion s'articule principalement sur la construction du modèle d'OL-Graph et l'implémentation de l'algorithme d'alignement sous forme d'un processus itératif. L'implémentation s'est concrétisée par la mise au point du système OLA pour l'alignement des ontologies OWL-Lite. L'expérimentation et l'analyse des résultats ont montré la validité du modèle de calcul de similarité : le modèle produit des alignements satisfaisants. Il est aussi extensible et permet d'intégrer d'autres éléments d'ontologies OWL-Lite pour un calcul de similarité plus fin. L'analyse a révélé de même la possibilité d'améliorer davantage le modèle de calcul de similarité en prenant en considération le calcul de la similarité inter-catégories d'entités.

Dans le chapitre qui suit, nous concluons sur ce mémoire par une récapitulation des points majeurs abordés. Nous ouvrons eventullement la voix de la recherche sur l'alignement d'ontologies par des propositions de pistes de recherches à moyen et à long terme.

Chapitre 6

Conclusion et perspectives

Par la présentation de ce mémoire, nous avons abordé le sujet d'alignement des ontologies OWL-Lite, un sujet qui connaît un dynamisme effervescent au sein de la communauté du Web sémantique. En effet, les ontologies constituent désormais la façon la mieux adaptée pour décrire les ressources sur le Web, le partage de la connaissance et la construction d'un Web doté de composants intelligents. Par conséquent, le besoin à d'outils d'alignement d'ontologies est d'une importance remarquable. Ces outils permettront d'assurer l'intéropérabilité d'ontologies face à la réalité d'un Web très hétérogène.

6.1 Conclusion

Pour s'inscrire dans un cadre de recherche qui se penche sur la question d'alignement d'ontologies sur le Web, nous avons présenté une méthode d'alignement d'ontologies décrites sous OWL-Lite. La méthode se base sur les principes suivants :

- une définition d'un modèle de graphe dit OL-Graph qui reflète par ces artéfacts les entités de l'ontologie,
- un modèle de similarité entre les entités d'ontologies par catégories,
- un processus itératif de calcul de celle-ci.

L'approche s'appuie sur une exploitation maximale d'information structurelle, terminologique et contextuelle pour le calcul de similarité. Le modèle de similarité se base sur l'OL-Graph qui, par ces artéfacts reflète toutes les entités et les relations qui existent dans une ontologie. Le modèle de similarité définit pour chaque couple d'entité d'une catégorie donnée une fonction d'agrégation. A partir de ce modèle, nous avons déduit un processus de calcul itératif qui résout, avec succès, le problème d'interdépendances entre les couples d'entités. Le modèle dérivé donne lieu à un système d'équations quasi-linéaires dont chacune représente la formule de calcul de similarité d'un couple d'entités. La résolution de ce système fournit à la fin les listes des couples d'entités avec leurs similarités maximales. A base de ces listes, des appariements sont ensuite produits.
La mise en œuvre de notre approche a donné naissance au système OLA [Euzenat et al., 2004b] pour la visualisation et l'alignement des ontologies OWL-Lite. Pour mettre à l'épreuve notre système nous avons participé à la compétition sur l'alignement d'ontologies EON. L'événement nous a fourni l'occasion de valider et vérifier les différents aspects du système.
Pour conclure, l'approche a montré des qualités comme :

- l'automaticité : OLA fournit deux modes d'exécution ; un premier totalement automatique : les paramètres de l'algorithme (poids, focntion de similarité léxicale etc.) sont lus à partir des fichiers XML, et un autre mode semi-automatique dans lequel l'utilisateur spécifie les paramètres requis.
- l'extensibilité : d'autres facteurs peuvent être facilement intégrés au calcul de similarité. A savoir les collections, les types de données composés. Le besoin à ces extensions se manifeste particulièrement dans le cas des ontologies décrites sous le langage OWL-DL ou OWL-Full.
- la possibilité de traiter le problème de circularité dans la définition des entités.

Cependant, malgré les résultats encourageants qu'a donné la méthode et les caractéristiques intéressantes qu'elle a fait preuve, des améliorations sont nécessaires pour atteindre une maturité complète.

6.2 Perspectives

A la lumière des résultats obtenus, on a pu dégagé des besoins constituant des améliorations à entreprendre à court terme. Leur implémentation permettra l'obtention de résultats d'alignement plus pertinents. D'autres besoins, s'inscrivent dans des projets de recherche à moyen et long terme : des pistes de recherche qui portent sur l'extension, la généralisation du modèle de calcul de similarité et l'aide à l'automatisation du choix optimal des paramétres de l'alignement.

6.2.1 Besoins à cours terme

Ces besoins peuvent être résumés en trois points :

- Premièrement, il serait très avantageux d'utiliser des techniques automatiques pour déterminer des patrons quand au choix des meilleurs poids associés au calcul de la similarité de chaque catégorie d'entité. Ceci permettra d'obtenir d'une part des alignements plus pertinents et d'autre part dispenser l'utilisateur du choix manuel et arbitraire de ces valeurs. Les techniques d'apprentissage machine sont très adaptées à ce genre de problèmes.
- Deuxièmement, le calcul de similarité s'effectue jusqu'à présent entre les entités de même catégorie. Toutefois, les tests ont montré que les résultats d'alignement pourraient être plus intéressants si on prend en considération le calcul de la similarité inter-catégories. Par conséquent, une adaptation de l'algorithme est nécessaire : considérer par exemple, le calcul de similarité entre les propriétés de nature "objet" et de nature "type de donnée" améliorera notablement les performances de notre méthode d'alignement.

  Finalement, affiner le calcul de similarité terminologique constitue un aspect très important pour améliorer la pertinence de résultat de l'algorithme. En effet, l'utilisation de l'API JWNL pour le calcul de la similarité terminologique s'avère de grande utilité pour définir des formules de calcul de similarité plus pointue entre les URIs des entités. Dans un premier temps, nous avons eu de bons résultats pour les cas des noms d'entités sensés (noms qui existent dans le dictionnaire). Cependant, les cas des noms combinés ou composés de chaînes de caractères aléatoires, n'affichent pas des résultats aussi satisfaisants. Il serai intéressant de penser à des méthodes plus efficaces pour mesurer la similarité entre des noms ayant des nomenclatures diversifiées.

6.2.2 Pistes de recherche à long terme

Nous avons dégagé des pistes de recherches, trois nous ont paru intéressantes à long terme :

- Une première piste vise à généraliser l'approche pour aligner les ontologies décrites en langage OWL-DL ou OWL-Full. Un tel besoin est imposé d'un côté par le fait que les ontologies pu-

bliées sur le Web ne sont pas toutes décrites en OWL-Lite. Elles feront nécessairement usage aux constructeurs de ces deux langages pour décrire des domaines divers. Pour se faire, une première idée consiste d'une part à étendre le modèle d'OL-Graph pour refléter tous les éléments des deux versions du langage OWL (DL et Full) et d'autre part adapter le modèle de similarité pour inclure d'autres types d'entités comme les collections, les types de données composés et les entités anonymes.

- Une deuxième piste consiste à penser à d'autres formules de calcul de similarité pour un rapprochement plus efficace de calcul de similarité sémantique. Bien que la formule proposée eut donné de bons résultats, la capture de l'information sémantique reste toujours incomplète. En plus, le potentiel sémantique est encore à épuiser, extraire et exploiter par le modèle de la similarité.
- Une troisième piste s'intéresse à l'aspect métricité : Il s'agit d'un côté de définir des métriques de qualité logiciel pour juger de la qualité des outils d'alignement et d'un autre côté définir une liste d'ontologies (benchmark) qui servira de test et d'évaluation de la qualité d'alignements produits.

Annexe A

Conception et organisation du projet

A.1 Considérations liées à la construction d'OL-Graph

Afin de construire la structure d'OL-Graph d'une manière à ce qu'il reflète toutes les entités de l'ontologie, certaines considérations devraient êtres prises en compte. Ces dernières portent d'une part sur l'interprétation de certains constructeurs d'OWL-Lite et d'autre part sur la représentation et le nommage des individus et des classes anonymes.

A.1.1 OL-Graph : vue d'implémentation

Le figure FIG. A.1 donne la structure d'OL-Graph de point de vue implémentation, les artefacts du graphe (les classes, arcs) sont nommés par les noms de classes JAVA qui les implémentant. Le sens des flèches des arcs indique le sens de navigation dans le graphe.

A.1.2 Représentation des classes anonymes

Les classes anonymes ne sont pas définies comme des classes indépendantes dans la grammaire d'OWL. En effet, elles font toujours partie de la définition d'une autre description (classe, domaine d'une propriété etc). Elles ne font qu'encapsuler alors un ensemble d'entités (descriptions dans le langage OWL). Ainsi, leur représentation dans l'OL-Graph respecte les hypothèses suivantes.

- Une classe anonyme encapsulant que des classes explicites sera toujours aplatie. Les classes contenues dans la description de la classe anonyme y seront liées directement.

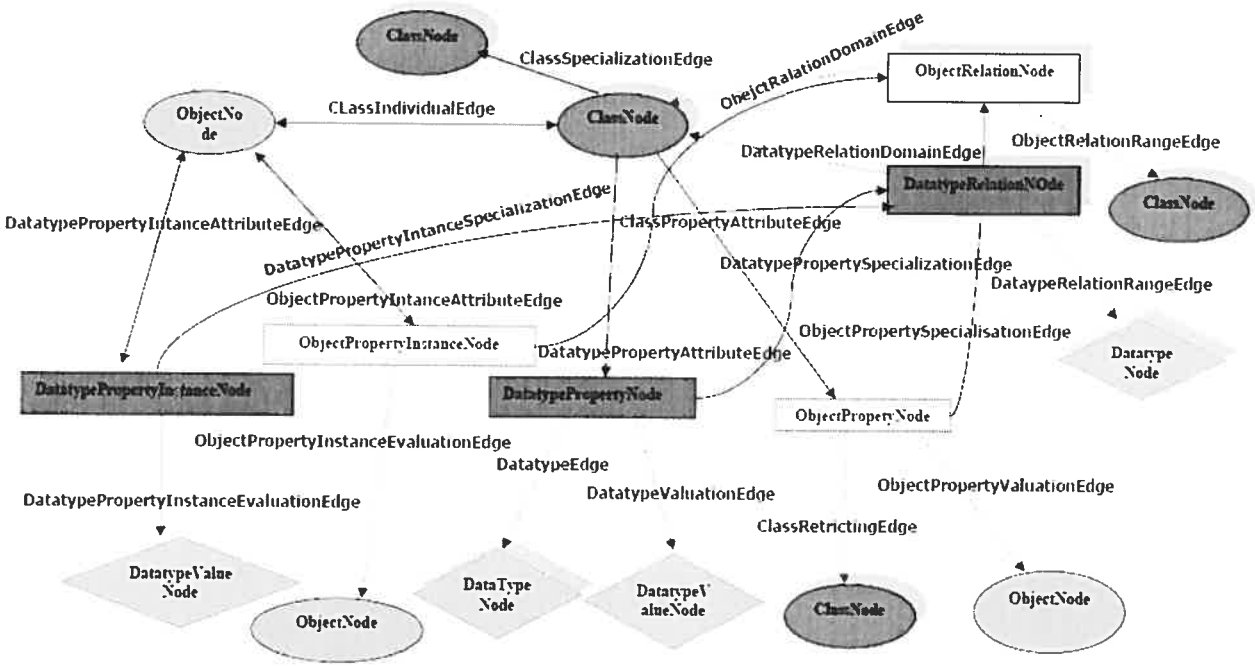

FIG. A.1 – OL-Graph : Les noms des classes JAVA des différents artefacts

- Une classe anonyme encapsulant une ou plusieurs propriétés ne sera pas aplatie par contre. Toutefois, une classe explicite correspondante à la classe anonyme sera crée[1] ayant comme identifiant la concaténation des identifiants des entités encapsulées.

A.1.3 Représentation des individus anonymes

Comme pour le cas des classes anonymes, les individus anonymes sont des entités XML (balises) déclarées au sein d'autres entités OWL sans avoir un identifiant. Du moment que ces individus ne sont pas des instances des classes explicites, ils ne seront liés à aucun nœud de type classe du graphe, en outre leur apport dans le calcul de similarité est faible, par conséquent ils ne sont pas représentés par l'OL-Graph.

A.1.4 Représentation des collections

Quoi qu'elles ne soient pas permises par OWL-Lite, les collections[2] sont prises en considération par la structure d'OL-Graph. Ainsi, la collection est traitée comme une classe anonyme ayant comme description l'intersection des entités de la collection.

A.1.5 Interprétation du lien d'intersection

Un lien d'intersection est traité comme un lien de spécialisation entre les classes. Ainsi, une classe *ClassA*, définie comme l'intersection de deux classes *ClassB* et *ClassC* sera considérée comme une sous classe des deux classes en question.

A.1.6 Liste des classes implémentant les nœuds

La table TAB. A.1 donne la liste des classes implémentant les nœuds d'OL-Graph.

A.1.7 Liste des classes implémentant les arcs

Le table TAB. A.3 donne la liste des classes implémentant les arcs.

[1]Les classes anonymes nouvellement crées lors du processus de construction du OL-Graph ne seront pas des candidats dans l'opération d'alignement par contre leur définition sera tenue en considération lors du calcul de similarité.

[2]Les collections sont introduites par le constructeur *unionOf*, qui ne fait pas parti des constructeurs OWL-Lite.

NOM DE LA CLASSE JAVA	DESCRIPTION
AbstarctGraphNode	Le nœud racine d'OL-Graph.
OWLOntologyNode	implémente la racine d'une ontologie (OL-Graph)
ClassNode	implémente un nœud correspondant à une classe explicite ou anonyme d'une ontologie
ObjectNode	implémente une instance d'une classe d'une ontologie
DatatypeNode	implémente un nœud de "type de donnée" (un prémitif XMLSchéma (String, Integer, boolean..)
DatatypeValueNode	implémente une valeur d'un "type de donnée"
ObjectPropertyNode	implémente une restriction de nature "objet" exprimée par rapport à une classe
ObjectRelationNode	implémente un nœud correspondant à une relation stand-alone de nature "objet"
DatatypeRelationNode	implémente un nœud correspondant à une relation stand-alone de nature "type de donnée"
DatatypePropertyNode	implémente une restriction de nature "type de donnée" exprimée par rapport à une classe
ObjectPropertyInstanceNode	implémente une instance de propriété de nature "objet" exprimée par rapport à un individu
DatatypePropertyInstanceNode	implémente une instance de propriété de nature "type de donnée" exprimée par rapport à un individu

TAB. A.1 – Table des classes JAVA implémentant les nœuds

A.1.8 Diagrammes UML d'OL-Graph

L'OL-Graph est implémenté par les classes du package *ca.umontreal.olgraph*. Le package regroupe les classes JAVA implémentant les nœuds et les arcs d'OL-Graph. Les digrammes de classes sont présentés respectivement par les deux figures FIG. A.3 et FIG. A.2.

A.1.9 Exemple de construction d'OL-Graph : cas d'une classe

Pour illustrer le processus de construction d'OL-Graph, nous présentons un exemple de construction de la partie d'OL-Graph correspondante à l'exemple ci-dessous. La construction commence par le parsing de la syntaxe OWL-Lite, l'aplatissement de la structure du graphe produite et la complétion finale d'OL-Graph.

A.1.9.1 Exemple d'une classe sous OWL-Lite

L'exemple[3] ci-dessous donne la définition de la classe *KoalaWithPhD* et la propriété nommée *hasHabitat* selon OWL-Lite. L'exemple sert à illustrer les trois phases de construction d'OL-Graph. La figure FIG A.4 montre les trois graphes résultatsf.

Exemple de la classe décrite en OWL-Lite

```
<owl:Class rdf:ID="KoalaWithPhD">
 <owl:versionInfo>1.2</owl:versionInfo>
 <owl:equivalentClass>
  <owl:Class>
   <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
     <owl:hasValue>
      <Degree rdf:ID="PhD" />
     </owl:hasValue>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#hasDegree" />
     </owl:onProperty>
    </owl:Restriction>
```

[3] l'exemple est tiré de l'ontologie Koala http ://protege.stanford.edu/plugins/owl/owl-library/koala.owl.

NOM DE LA CLASSE JAVA	DESCRIPTION
AbstractEdge	lien abstrait (entre deux nœuds)
ClassSpecialisationEdge	lien de spécialisation entre deux classes
ObjectRelationRangeEdge	lien entre une relation stand-alone de nature "objet" et une classe (range)
ObjectRelationDomainEdge	lien entre une relation stand-alone de nature "objet" et une classe (domain)
DatatypeRelationRangeEdge	lien entre une relation stand-alone de nature "type de donnée" et une classe (range)
DatatypeRelationDomainEdge	lien entre une relation stand-alone de nature "type de donnée" et une classe (domaine)
ObjectPropertyValuationEdge	lien de valuation entre une propriété de nature "objet" et une valeur objet
ObjectPropertyAttributeEdge	lien d'attribution entre une propriété de nature "objet" à une classe
DatatypePropertyAttributeEdge	lien d'attribution d'une propriété de nature "type donnée" à une classe
DatatypePropertyValuationEdge	lien de valuation entre une propriété de nature "type de donnée" et une valeur de donnée

TAB. A.2 – Table des classes JAVA implémentant les arcs d'OL-Graph : partie 1

NOM DE LA CLASSE JAVA	DESCRIPTION
DatatypeEdge	lien d'attribution entre une propriété de nature "type de donnée" et un prémitif de XMLSchema
ClassIndividualEdge	lien entre une classe et un individu
ObjectPropertyValuationEdge	lien de valuation entre une propriété de nature "objet" et une valeur d'objet
ObjectPropertySpecializationEdge	lien de spécialisation entre une propriété de nature "objet" et une relation stand-alone du même nature
DatatypePropertySpecializationEdge	lien de spécialisation entre une propriété de "type de donnée" et une relation stand-alone du même nature
ObjectPropertyInstanceAttributeEdge	line d'attribution entre une instance de propriété de nature "objet" et un individu
ObjectPropertyInstanceValuationEdge	lien de valuation entre une instance de propriété de nature "objet" et une valeur d'objet
DatatypePropertyInstanceValuationEdge	lien de valuation entre une instance de propriété de nature "type de donnée" et une valeur de donnée
DatatypePropertyInstanceAttributeEdge	lien d'attribution entre une instance de propriété de nature "type de donnée" et un individu
ObjectPropertyInstanceSpecializationEdge	lien de spécialisation entre une instance de propriété de nature "objet" et une relation stand-alone du même nature
DatatypePropertyInstanceSpecializationEdge	implémente un lien de spécialisation entre une instance de propriété "type de donnée" et une relation stand-alone du même nature

TAB. A.3 – Table des classes JAVA implémentant les arcs d'OL-Graph : suite

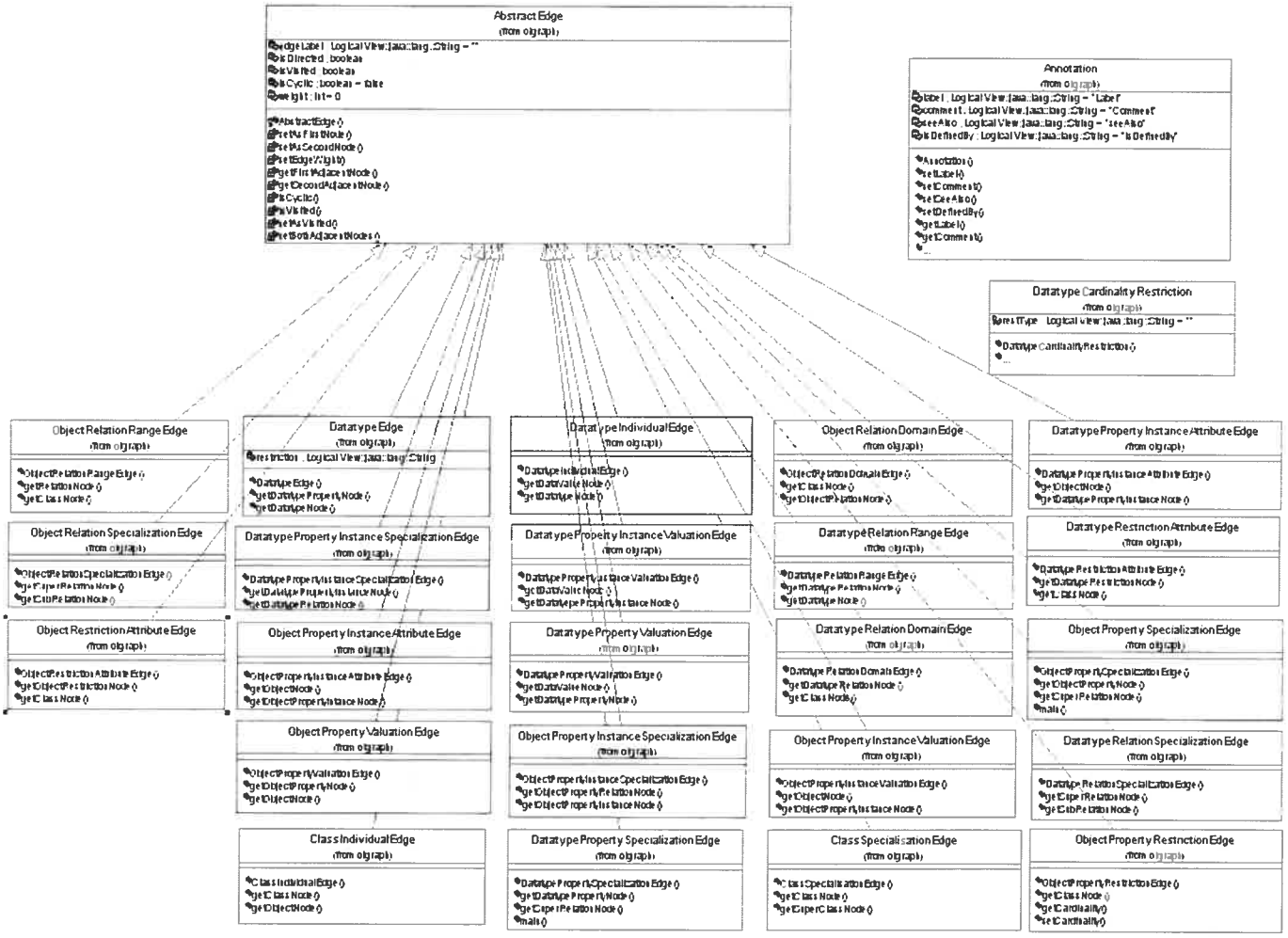

FIG. A.2 – Diagramme UML de classes d'OL-Graph : les arcs du graphe

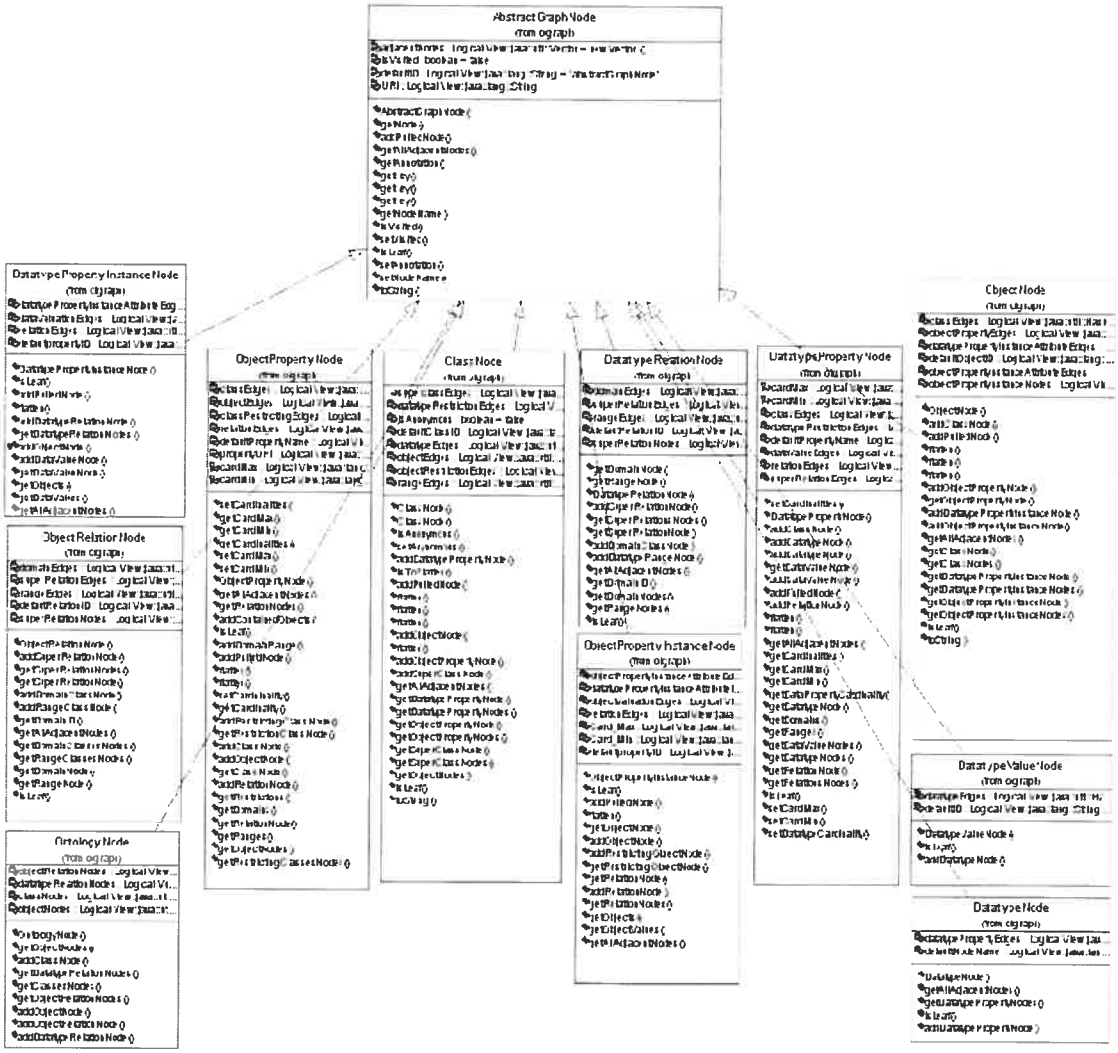

FIG. A.3 – Diagramme UML de classes d'OL-Graph : les nœuds du graphe

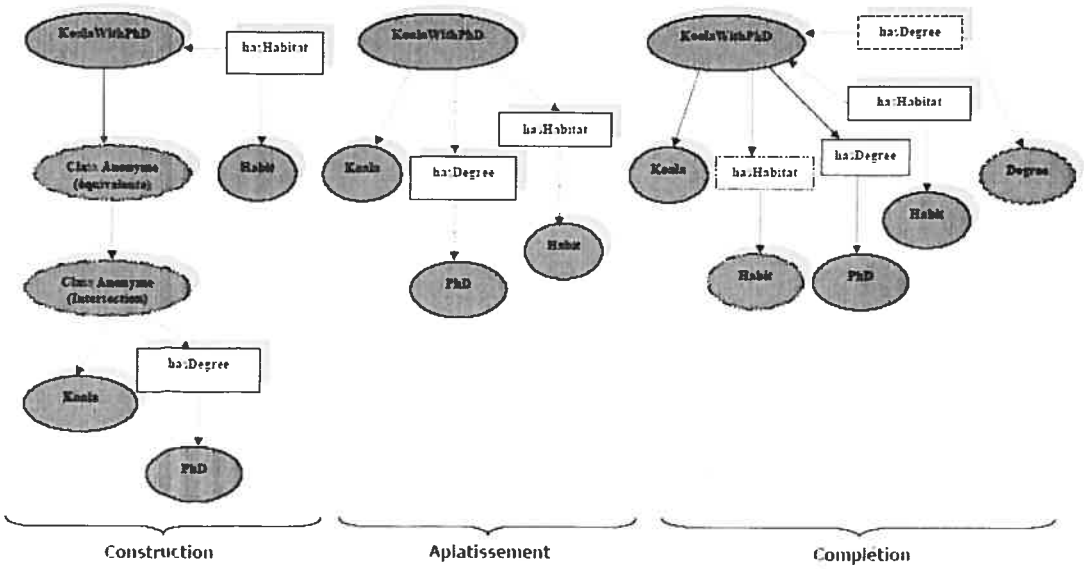

FIG. A.4 – Processus de construction d'OL-Graph : exemple précédent

```
    <owl:Class rdf:about="#Koala" />
   </owl:intersectionOf>
  </owl:Class>
 </owl:equivalentClass>
</owl:Class>

 <owl:ObjectProperty rdf:ID="hasHabitat">
    <rdfs:domain rdf:resource="#KoalaWithPhD"/>
    <rdfs:range rdf:resource="#Habitat"/>
  </owl:ObjectProperty>
```

- La classe *KoalaWithPhD* est définie comme une classe équivalente à une classe anonyme dont les instances sont l'intersection des instances de la restriction exprimée par la propriété *hasDegree* ayant la valeur *PhD* et la classe *Koala*.

- La propriété *hasHabitat* a comme domaine la classe *KoalaWithPhD* et le range la classe *Habitat*.

A.1.9.2 Construction

La construction consiste à parser la syntaxe OWL-Lite et construire les artéfacts du graphe OL-Graph. La construction résulte parfois en un ajout de nœuds associés aux classes anonymes nécessitant un aplatissement. Le parsing et la construction initiale d'OL-Graph pour l'exemple précédent est présentée par la figure FIG. A.4. On note particulièrement l'ajout des nœuds de classes correspondantes aux classes anonymes et la création de la relation stand-alone correspondante à la propriété *hasHabitat*. Le premier graphe de la figure FIG A.4 représente le résultat de l'opération de construction.

A.1.9.3 Aplatissement

L'aplatissement consiste essentiellement à supprimer les nœuds correspondants aux entités anonymes. Les règles qui régissent un aplatissement sont les suivantes :

Toutes les entités descendantes d'une classe anonyme devraient être attachées au nœud (parent de la classe anonyme) sauf pour les cas suivants :

- Le parent est une classe anonyme à son tour.
- Le parent est un nœud représentant une propriété alors que la classe anonyme est liée respectivement soit à plus de deux classes, soit à une classe et un autre nœud (quelque soit son type).

Le deuxième graphe de la figure FIG. A.4 montre le résultat de l'aplatissement.

A.1.9.4 Complétion

La complétion de l'OL-Graph de l'exemple précédent donne le troisième graphe de la FIG. A.4. On remarque l'instantiation de la relation stand-alone de nature "objet" donnant lieu à une restriction exprimée par une propriété dérivée de la relation stand-alone. Le range de la restriction est le même que celui de la relation stand-alone. La complétion de l'OL-Graph porte aussi sur la création des nœuds correspondants aux instances de propriétés.

A.1.10 Résultats de l'exemple de test

On présente la suite des résultats de calcul de similarité de l'exemple 4.3.6. Les résultats concernent les relations de nature "objet" et les relations de nature "type de donnée".

COUPLE	SIMILARITÉ LEXI.	ITÉR. 0	ITÉR. 1	ITÉR. 2	ITÉR. 3
(rooms, owner)	0.2000	0.3294	0.3326	0.3340	0.3340
(rooms, locomotion)	0.2667	0.3816	0.3860	0.3870	0.3870
(dwelling, owner)	0.1538	0.2639	0.2684	0.2701	0.2701
(dwelling, locomotion)	0.1111	0.4079	0.4158	0.4181	0.4181
(spouse, owner)	0.6250	0.5296	0.5344	0.5360	0.5360
(spouse, locomotion)	0.1250	0.3296	0.3344	0.3360	0.3360

TAB. A.4 – Table des similarités entre les relations de nature "objet"

A.1.10.1 Similarités entre relations de nature "objet"

La table TAB. A.4 donne les similarités des relations de nature "objet" de l'exemple présenté à la section 4.3.6.

A.1.10.2 Similarités entre relations de nature "type de donnée"

La table TAB. A.5 donne les similarités des relations de nature "type de donnée" de l'exemple présenté à la section 4.3.6.

A.2 Exemple d'alignement

Le fichier XML ci-dessus est un exemple d'alignement, produit par le système OLA, d'ontologies de l'exemple 4.3.6. Le format du fichier respecte un schéma bien spécifique afin de simplifier leur utilisation automatique par d'autres outils d'alignement. Les élements du schéma sont décrits comme suit :

- **classe *Alignment*** : décrit un alignement bien particulier. Il a les propriétés suivantes
  - xml : prend une valeur boolean pour indiquant si l'alignement peut être lu suivant le DTD,
  - level : prend une des valeurs "0", "1" ou "2OWL" indiquant le niveau de l'alignement,
  - type : indiquant le type d'alignement,
  - uri1 : l'URI de la première ontologie,
  - uri2 : l'URI de la deuxième ontologie,
  - map : prend une valeur de (Cell*) indiquant une correpondance entre deux entités
- **classe *Cell*** : il a les éléments suivants

COUPLE	SIMILARITÉ LEXI.	ITÉR. 0	ITÉR. 1	ITÉR. 2	ITÉR. 3
(age, model)	0.2500	0.4775	0.4777	0.4784	0.4784
(age, name)	0.2857	0.5858	0.5904	0.5914	0.5914
(age, salary)	0.2222	0.5756	0.5803	0.5813	0.5813
(addres, model)	0.1667	0.5434	0.5466	0.5480	0.5480
(address, name)	0.6091	0.6531	0.6575	0.6585	0.6585
(address, salary)	0.1538	0.4558	0.4601	0.4612	0.4612
(surface, model)	0.3750	0.6002	0.6002	0.6002	0.6002
(surface,, name)	0.2000	0.4374	0.4374	0.4374	0.4374
(surface,, salary)	0.1538	0.4342	0.4342	0.4342	0.4342
(name, model)	0.3817	0.5455	0.5456	0.5463	0.5463
(name, name)	1.0000	0.8867	0.8914	0.8924	0.8924
(name, salary)	0.2000	0.5515	0.5562	0.5571	0.5571

TAB. A.5 – Table des similarités entre les relations de nature "type de donnée"

- entity1 : l'URI de la première entité du couple aligné,
- enitty2 : l'URI de la deuxième entité du couple aligné,
- measure : la similarité calculée,
- relation : la classe relation.

– **classe *Relation*** : la classe relation représente le type de relation déduite entre la première entité de l'ontologie et la deuxième. Cette classe n'est spécifiée dans ce format (elle est indiquée par une chaîne de caractères $=$, $>$, $<$).

Exemple d'alignement

```
<?xml version='1.0' encoding='utf-8' standalone='no'?> <rdf:RDF
xmlns='http://knowledgeweb.semanticweb.org/heterogeneity/alignment'
         xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:xsd='http://www.w3.org/2001/XMLSchema#'>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type></type>
  <uri1>file:/R:/ontoTest/my-onto-1.owl</uri1>
  <uri2>file:/R:/ontoTest/my-onto-2.owl</uri2>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#address'/>
      <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#name'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8831168831168831</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#age'/>
      <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#salary'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8703703703703703</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#spouse'/>
      <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#locomotion'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.9125</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#Room'/>
      <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#Person'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8666666666666667</measure>
      <relation>=</relation>
    </Cell>
  </map>
```

```
    <map>
      <Cell>
        <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#Human'/>
        <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#Person'/>
        <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8636363636363636</measure>
        <relation>=</relation>
      </Cell>
    </map>
    <map>
      <Cell>
        <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#rooms'/>
        <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#locomotion'/>
        <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.9266666666666666</measure>
        <relation>=</relation>
      </Cell>
    </map>
    <map>
      <Cell>
        <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#dwelling'/>
        <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#locomotion'/>
        <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.9111111111111111</measure>
        <relation>=</relation>
      </Cell>
    </map>
    <map>
      <Cell>
        <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#surface'/>
        <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#name'/>
        <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8831168831168831</measure>
        <relation>=</relation>
      </Cell>
    </map>
    <map>
      <Cell>
        <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#name'/>
        <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#name'/>
        <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
        <relation>=</relation>
      </Cell>
    </map>
    <map>
      <Cell>
        <entity1 rdf:resource='file:/R:/ontoTest/my-onto-1.owl#Flat'/>
        <entity2 rdf:resource='file:/R:/ontoTest/my-onto-2.owl#Person'/>
        <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.8333333333333334</measure>
        <relation>=</relation>
      </Cell>
    </map>
  </Alignment> </rdf:RDF>
```

A.3 Exemples d'ontologies en OWL

On présente ci-dessous les deux ontologies *onto1.owl* et *onto2.owl* décrites sous le langage OWL-Lite et servies pour le test de l'exemple 4.3.6. Les deux ontologies sont fictives. Ainsi, pour des raisons de simplification, elles ne font pas figurer tous les constructeurs du langage OWL-Lite.

A.3.1 Première ontologie OWL de l'exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE rdf:RDF (View Source for full doctype...)>
<rdf:RDF
   xmlns="http://co4.inrialpes.fr/align/Contest/104/onto.rdf#"
   xmlns:units="http://visus.mit.edu/fontomri/0.01/units.owl#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:ical="http://www.w3.org/2002/12/cal/#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:wot="http://xmlns.com/wot/0.1/"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:dctype="http://purl.org/dc/dcmitype/">

 <owl:Class rdf:ID="Human">
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="#dwelling" />
       <owl:allValuesFrom rdf:resource="#Flat" />
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#spouse" />
         <owl:allValuesFrom rdf:resource="#Human" />
       </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#spouse" />
         <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1
         </owl:maxCardinality>
       </owl:Restriction>
    </rdfs:subClassOf>
 </owl:Class>

 <owl:Class rdf:ID="Flat">
   <rdfs:label xml:lang="en">Flat</rdfs:label>
     <rdfs:subClassOf>
      <owl:Restriction>
       <owl:onProperty rdf:resource="#rooms" />
       <owl:allValuesFrom rdf:resource="#Room" />
```

```
      </owl:Restriction>
    </rdfs:subClassOf>
 </owl:Class>

 <owl:Class rdf:ID="Room">
   <rdfs:label xml:lang="en">Room</rdfs:label>
 </owl:Class>

<owl:DatatypeProperty rdf:ID="name">
 <rdfs:domain rdf:resource="#Human" />
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
 <rdfs:label xml:lang="en">key</rdfs:label>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="age">
 <rdfs:domain rdf:resource="#Human" />
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger" />
 </owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="address">
 <rdfs:domain rdf:resource="#Flat" />
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
 <rdfs:label xml:lang="en">key</rdfs:label>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="surface">
 <rdfs:domain rdf:resource="#Room" />
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger" />
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="dwelling">
   <rdfs:domain rdf:resource="#Human" />
   <rdfs:range rdf:resource="#Flat" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="spouse">
   <rdfs:domain rdf:resource="#Human" />
   <rdfs:range rdf:resource="#Human" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="rooms">
   <rdfs:domain rdf:resource="#Flat" />
   <rdfs:range rdf:resource="#Room" />
</owl:ObjectProperty>

</rdf:RDF>
```

A.3.2 Deuxième ontologie OWL de l'exemple

```
 <?xml version="1.0" encoding="utf-8" ?>
 <!DOCTYPE rdf:RDF (View Source for full doctype...)>
<rdf:RDF
  xmlns="http://co4.inrialpes.fr/align/Contest/104/onto.rdf#"
```

```
  xmlns:units="http://visus.mit.edu/fontomri/0.01/units.owl#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:ical="http://www.w3.org/2002/12/cal/#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:wot="http://xmlns.com/wot/0.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:dctype="http://purl.org/dc/dcmitype/">

<owl:Class rdf:ID="Person">
 <rdfs:subClassOf>
  <owl:Restriction>
      <owl:onProperty rdf:resource="#locomotion" />
      <owl:allValuesFrom rdf:resource="#Car" />
  </owl:Restriction>
</rdfs:subClassOf>

</owl:Class>
  <owl:Class rdf:ID="Car">
  <rdfs:label xml:lang="en">Car</rdfs:label>
  <rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty rdf:resource="#owner" />
      <owl:allValuesFrom rdf:resource="#Person" />
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="name">
   <rdfs:domain rdf:resource="#Person" />
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
   <rdfs:label xml:lang="en">key</rdfs:label>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="salary">
   <rdfs:domain rdf:resource="#Person" />
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="model">
   <rdfs:domain rdf:resource="#Car" />
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
   <rdfs:label xml:lang="en">key</rdfs:label>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="locomotion">
   <rdfs:domain rdf:resource="#Person" />
   <rdfs:range rdf:resource="#Car" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="owner">
  <rdfs:domain rdf:resource="#Car" />
```

```
    <rdfs:range rdf:resource="#Person" />
  </owl:ObjectProperty>
</rdf:RDF>
```

Bibliographie

[Baader et al., 2003] Baader, F., Horrocks, I., et Sattler, U. (2003). Description Logics as Ontology Languages for the Semantic Web, *http ://citeseer.ist.psu.edu/baader03description.html*.

[Baader et Nutt, 2002] Baader, F. et Nutt, W. (2002). *Basic Description Logic*, http ://www.inf.unibz.it/ franconi/dl/course/dlhb/dlhb-02.pdf.

[Bach et al., 2004] Bach, T. L., Dieng-Kuntz, R., et Gandon., F. (2004). On ontology Problems for Building Semantic Web in Corporate Multi-communities Organisation. In *Proceeding of ICEIS*, pages 236–243, *http ://dblp.uni–trier.de*.

[Bechhofer et al., 2003] Bechhofer, S., Volz, R., et Lord, P. (2003). Cooking the Semantic Web with the OWL API. In *Proceedings of the ISWC Sanibel Island*, pages 14–32, *http ://citeseer.ist.psu.edu/bechhofer03cooking.html*, Florida, USA.

[Berlin et Motro, 2002] Berlin, J. et Motro, A. (2002). Database Schema Matching Using Machine Learning with Feature Selection. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE 02)*, pages 452–466, *http ://portal.acm.org/citation.cfm*, London, UK. Springer-Verlag.

[Bouquet et al., 2003] Bouquet, P., Serfani, L., et Zanobni, S. (2003). Semantic Coordination : New Approach and an Application. In *Proceedings of the 11th international Conference on World Wide Web ISWC*, pages 130–145, *http ://citeseer.lcs.mit.edu/676539.html*.

[Chandrasekaran et al., 1999] Chandrasekaran, B., Josephson, J. R., et Benjamins, V. R. (1999). What Are Ontologies, and Why Do We Need Them ?,. pages 20–26, *http ://dx.doi.org/10.1109/5254.747902*.

[Cohen et al., 2003] Cohen, W., Ravikumar, P., et Fienberg, S. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the IJCAI*, pages *http ://www.cs.cmu.edu/ wcohen/postscript/ijcai–ws–2003.pdf*.

[Connolly et al., 2001] Connolly, D., Harmelen, F. V., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., et Stein, L. A. (2001). DAML-OIL Guide Reference. *http ://www.w3c.org/TR/daml+oil-reference*.

[Cui et al., 2003] Cui, Z., Shepherdson, J. W., et Li, Y. (2003). An ontology-Based Approach to eCatalogue Management. pages 76–83, *http ://dx.doi.org/10.1023/A :1027383504596*.

[Decker et al., 2000] Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M. C. A., Broekstra, J., Erdmann, M., et Horrocks, I. (2000). The Semantic Web : The Roles of XML and RDF. *IEEE Internet Computing*, pages 63–74, *http ://citeseer.ist.psu.edu/decker00semantic.html*.

[Dieng et Hug, 1998] Dieng, R. et Hug, S. (1998). Comparison of Personal Ontologies Represented through Conceptual Graphs. In *Proceeding of ECAI*, pages 341–345, *http ://dblp.uni–trier.de*.

[Do et Rahm, 2002] Do, H. et Rahm, E. (2002). COMA : A System for Flexible Combination of Schema Matching Approaches, *http ://citeseer.ist.psu.edu/do02coma.html*.

[Doan et al., 2003] Doan, A., Domingos, P., et Halevy, A. (2003). Learning to Match the Schemas of Data Sources : A Multistrategy Approach. pages 279–301, *http ://dx.doi.org/10.1023/A :1021765902788*.

[Doan et al., 2001] Doan, A., Domingos, P., et Halevy, A. Y. (2001). Reconciling Schemas of Disparate Data sources : A Machine-Learning Approach. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 509–520, *http ://doi.acm.org/10.1145/375663.375731*, New York,USA. ACM Press.

[Doan et al., 2002] Doan, A., Madhavan, J., Domingos, P., et Halevy, A. (2002). Learning to Map between Ontologies on the Semantic Web. In *Proceedings of the 11th International Conference on World Wide Web*, pages 662–673, *http ://doi.acm.org/10.1145/511446.511532*. ACM Press.

[Ehrig et Sure, 2004] Ehrig, M. et Sure, Y. (2004). Ontology Mapping an Integrated Approach. pages 76–91, *http ://citeseer.ist.psu.edu/article/ehrig04ontology.html*.

[EON, 2004] EON, G. (2004). EON Ontology Alignment Contest. In *Proceedings of the 3rd Evaluation of Ontology-based Tools (EON) Workshop, http ://co4.inrialpes.fr/align/Contest/*.

[Euzenat et al., 2004a] Euzenat, J., Bach, T. L., Barrasa, J., Bouquet, P., Bo, J. D., Dieng, R., Ehrig, M., Lara, R., Maynard, D., Napoli, A., Stamou, G., Stuckenschmidt, H., Shvaiko, P., Tessaris, S., Acker, S. V., et Zaihrayeu, I. (2004a). State of the Art on Ontology Alignment, pages 4–25, *www.starlab.vub.ac.be/research/projects/knowledgeweb/kweb-223.pdf*.

[Euzenat et al., 2004b] Euzenat, J., Loup, D., Touzani, M., et Valtchev, P. (November 2004b). Ontology Alignment with OLA. In *3rd International Workshop : Semantic Web Conference EON*, pages 341–371, *http ://www.iro.umontreal.ca/ owlola/pdf/align–compet–EON.pdf*.

[Euzenat et Valtchev, 2003] Euzenat, J. et Valtchev, P. (2003). An Integrative Proximity Measure for Ontology Alignment. In *Proceedings of the Workshop on Semantic Information Integration*, pages 33–38, *http ://citeseer.ist.psu.edu/euzenat03integrative.html.*

[Euzenat et Valtchev, 2004] Euzenat, J. et Valtchev, P. (2004). Similarity-based Ontology Alignment in OWL-Lite. In *Procedings of the 15th ECAI*, pages 333–337, *http ://www.iro.umontreal.ca/ owlola/pdf/align–ECAI04–FSub.pdf*, Valencia, Spain.

[Giunchiglia F. et Yatskevich, 2004] Giunchiglia F., P. S. et Yatskevich, M. (2004). S-Match : An Algorithm and an Implementation of Semantic Matching. In *Proceedings of ESWC*, pages 61–75, *http ://drops.dagstuhl.de/opus/volltexte/2005/37/pdf/04391.GiunchigliaFausto1.Paper.37.pdf.*

[Gruber, 1995] Gruber, T. R. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. pages 907–928, *http ://citeseer.ist.psu.edu/gruber93toward.html*, Duluth, USA. Academic Press, Inc.

[Jaakkola et al., 1999] Jaakkola, T., Diekhans, M., et Haussler, D. (1999). Using the fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 149–158. AAAI Press.

[Levenshtein, 1966] Levenshtein, I. V. (1966). Binary Codes Capables of Corrections, Deletions, Insertions and Reversals. In *Soviet Physics-Doklady 10*, pages 707–710.

[Li et Clifton, 1994] Li, W.-S. et Clifton, C. (1994). Semantic Integration in Heterogeneous Databases Using Neural Networks. In *Proceedings of the 20th International Conference on Very Large Data Bases VLDB*, pages 1–12, *www.vldb.org/conf/1994/P001.PDF*. Morgan Kaufmann Publishers Inc.

[Madhavan et al., 2001] Madhavan, J., Bernstein, P. A., et Rahm, E. (2001). Generic Schema Matching with Cupid. In *Proceedings of The VLDB Journal*, pages 49–58, *http ://citeseer.ist.psu.edu/madhavan01generic.html.*

[Magini et al., 2003] Magini, B., Serafani, L., et M.Speranza (2003). Making Explicit the Semantic Hidden in Schema Models. In *Proceedings of the Workshop on Human Language Technology for the Semantic Web and Web Services ISWC*, pages *http ://tcc.itc.it/people/speranza/iswc–2003.ps* .

[Melnik, 2002] Melnik, S. (2002). Similarity flooding : A Versatile Graph Matching Algorithm and its Application to Schema Matching. In *Proceedings of the 18th International Conference on Data Engineering ICDE02. http ://portal.acm.org/citation.cfm ?id=879024*, pages 117, *http ://portal.acm.org/citation.cfm ?id=879024*. IEEE Computer Society.

[Miller, 1995] Miller, G. A. (1995). Wordnet : a Lexical Database for English. pages 39–41, *http ://doi.acm.org/10.1145/219717.219748.*

[Noy et McGuinness, 2001] Noy, N. F. et McGuinness, D. L. (2001). Ontology Development 101 : A Guide to Creating Your First Ontology, *http ://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html*.

[Noy et Musen, 2001] Noy, N. F. et Musen, M. A. (2001). Anchor-PROMPT : Using Non-Local Context for Semantic Matching. In *Proceedings of the Workshop on Ontologies and Information Sharing IJCAI*, pages *www–smi.stanford.edu/pubs/SMI_Reports/SMI–2001–0889.pdf*.

[Ristad et Yianilos, 1998] Ristad, E. S. et Yianilos, P. N. (1998). Learning String-Edit Distance. pages 522–532, *http ://dx.doi.org/10.1109/34.682181*.

[Sekine et al., 1999] Sekine, S., Sudo, K., et Ogino, T. (1999). Statistical Matching of Two Ontologies, *http ://citeseer.ist.psu.edu/sekine99statistical.html*.

[Shasha et al., 2002] Shasha, D., Wang, J. T.-L., et Giugno, R. (2002). Algorithmics and Applications of Tree and Graph Searching. In *Proceeding of the Symposium on Principles of Database Systems*, pages 39–52, *http ://citeseer.ist.psu.edu/shasha02algorithmics.html*.

[Shasha et Zhang, 1997] Shasha, D. et Zhang, K. (1997). Approximate Tree Pattern Matching. In *Pattern Matching Algorithms*, pages 341–371, *http ://citeseer.ist.psu.edu/shasha95approximate.html*. Oxford University Press.

[Shutt et al., 1999] Shutt, D., Bernstein, P. A., Bergstraesser, T., Carlson, J., Pal, S., et Sanders, P. (1999). Microsoft Repository version 2 and the Open Information Model. pages 71–98, *http ://dx.doi.org/10.1016/S0306–4379(99)00006–X*.

[Smithv et al., 2004] Smithv, M. K., Welty, C., et McGuinness, D. (2004). Ontology Web language Guide, *http ://www.w3.org/TR/2003/PR-owl-guide-20031215/*, W3C.

[Stumme et Maedche, 2001] Stumme, G. et Maedche, A. (2001). FCA-MERGE : Bottom-Up Merging of Ontologies, pages 225–130, *http ://citeseer.ist.psu.edu/article/stumme01fcamerge.html*.

[Swartout et al., 1996] Swartout, B., Patil, R., Knight, K., et Russ, T. (1996). Toward Distributed Use of Large-Scale Ontologies.

[Valtchev, 1999] Valtchev, P. (1999). *Construction automatique de taxonomies pour l'aide à la représentation de connaissance par objets*. PhD thesis, Université de Grenoble 1.

[Valtchev et Euzenat, 1997] Valtchev, P. et Euzenat, J. (1997). Dissimilarity Measure for Collections of Objects and Values. In *IDA '97 : Proceedings of the Second International Symposium on Advances in Intelligent Data Analysis, Reasoning about Data*, pages 259–272, *http ://portal.acm.org/citation.cfm ?id=647965.741438*. Springer-Verlag.

[Valtchev et al., 2004] Valtchev, P., Touzani, et Loup, D. (2004). Ontology Lite Alignment, *www.iro.u.montreal.ca/ owlola*.

[Wiesman et al., 2002] Wiesman, F., Roos, N., et Vogt, P. (2002). Automatic Ontology Mapping for Agent Communication. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS*, pages 563–564, *http ://doi.acm.org/10.1145/544862.544876*. ACM Press.