UNIVERSITÉ DU QUÉBEC À MONTRÉAL

UNE MÉTHODE DE FILTRAGE DES TREILLIS DE CONCEPTS POUR LA RESTRUCTURATION D'ONTOLOGIES PAR L'ANALYSE RELATIONNELLE DE CONCEPTS

THÈSE

PRÉSENTÉE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN INFORMATIQUE

PAR

SCHAHRAZED FENNOUH

SEPTEMBRE 2016

DÉDICACE

À mes très chers parents, *Essaid* et *Fadila*

À mes très chers enfants, *Rayan* et *Racim*

Et à tous ceux qui me sont chers,

Je dédie ce travail...

REMERCIEMENTS

Je tiens à remercier les personnes qui m'ont accompagnée, encouragée et soutenue tout au long de cette thèse :

Tout d'abord, M. *Roger Nkambou*, mon directeur de recherche, et M. *Petko Valtchev*, mon co-directeur, pour m'avoir proposé ce sujet et avoir dirigé mes travaux, pour m'avoir prodigué les conseils judicieux et m'avoir donné des orientations et des éclairages sur la conduite de cette thèse, et enfin pour la patience qu'ils ont démontré à mon égard durant toutes ces années de doctorat.

J'exprime tous mes remerciements accompagnés de ma gratitude aux membres du jury : M. *Vladimir Makarenkov*, M. *Michel Gagnon* et M. *Éric Beaudry* pour avoir corrigé, examiné et critiqué mon travail. Leurs commentaires constructifs et leurs recommandations m'ont permis de grandement améliorer mon rapport.

Je remercie également M. *Amine Mohamed Rouane-Hacène* avec qui j'ai eu le grand plaisir de travailler sur le projet *Priows*. Tout d'abord, pour m'avoir transmis son savoir sur l'Analyse relationnelle de concepts, et ensuite pour ses précieux conseils et ses avis éclairés.

Ma profonde reconnaissance va spécialement à Melle. *Linda Zeghache* au sein de laquelle j'ai toujours trouvé le soutien et la compréhension dont j'avais besoin. Je la remercie pour l'aide précieuse qu'elle m'a toujours apportée, notamment sur les aspects techniques.

Mes vifs remerciements vont à mes amis de l'UQAM : *Othalia*, *Jamil* et *El-Mahdi* pour leur aide appréciable ainsi que pour leurs encouragements.

Je remercie le personnel du département d'informatique de l'UQAM pour n'avoir ménagé aucun effort pour me faciliter le bon déroulement de ce travail.

Un immense merci à mon mari, *Rédha*, qui a su m'apporter le support moral et matériel dont j'avais besoin. Je lui suis très reconnaissante pour sa présence, sa grande patience et son soutien constant tout au long de ce chemin ardu du doctorat.

Je suis également très reconnaissante envers ma famille en Algérie, mes soeurs et mon frère, pour leur amour et pour m'avoir accompagnée de leurs encouragements continus et de leur soutien moral agréable. Je remercie particulièrement *Ahlem* pour sa présence, sa disponibilité, son aide, et surtout pour avoir partagé avec moi les moments importants de cette thèse.

Je souhaite finalement adresser mes remerciements à mes chers parents pour avoir toujours cru en moi, pour leur soutien indéfectible et ininterrompu, et enfin pour leur amour et leur affection. Sans leurs prières et leur bénédiction, je n'aurais jamais pu avancer, surmonter les moments difficiles et arriver au bout de cette aventure. Que Dieu les garde !

Et pour être sûre de n'oublier personne, que tous ceux, qui de près ou loin, ont contribué, par leurs conseils, leurs encouragements ou leur amitié, à l'aboutissement de ce travail, trouvent ici l'expression de ma profonde reconnaissance.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

LISTE DES ACRONYMES

AFC : Analyse Formelle de Concepts

API : Application Programming Interface

ARC : Analyse Relationnelle de Concepts

ASMOV : Automated Semantic Matching of Ontologies with Verification

CMS : Conference Management System

CRUD : Create, Retrieve, Update and Delete

DAO : Drosophila Anatomy Ontology

DTD : Document Type Definition

FCR : Famille de Contextes Relationnels

FTR : Famille des Treillis Relationnels

GL : Génie Logiciel

IA : Intelligence Artificielle

IC : Ingénierie des Connaissances

IO : Ingénierie Ontologique

ISF : Integrated Semantic Framework

JRuby : Java based version of Ruby

LD : Logique de Description

OAEI : Ontology Alignment Evaluation Initiative

ODE : Ontology Design Environment

ODM : Ontology Definition Metamodel

OntoEdit : Ontology Editor

OWL : Ontology Web Language

OWL-DL : Ontology Web Language Description Logics

POMs : Probabilistic Ontology Models

PRIOWS : Projet de recherche en ingénierie ontologique et web sémantique

RDF : Resource Description Framework

RDF(S) : Resource Description Framework Schema

REA : Resource Event Agent

SBC : Systeme à base de connaissances

SPARQL : SPARQL Protocol and RDF Query Language

TEXCOMON : TEXt COncept Map ONtology

TOVE : TOronto Virtual Entreprise

UML : Unificd Modeling Language

XML : eXtensible Markup Language

RÉSUMÉ

Un modèle ontologique, comme la plupart des projets d'ingénierie des systèmes d'information (modèles conceptuels et code source), est sujet à des erreurs et des anomalies. Des travaux existants ont tenté de détecter ces anomalies et ont proposé des méthodes pour leur correction en passant par un processus de restructuration. Ce processus vise l'amélioration de la qualité d'une ontologie et par conséquent faciliter sa compréhension et sa maintenabilité. Des approches existantes ont abordé la restructuration dans différentes perspectives. Cependant, il n'existe toujours pas de méthodologie bien établie qui couvre tout le processus de restructuration et qui permet de faire un remaniement global de la structure de l'ontologie. De plus, ces approches se focalisent sur un seul niveau de la restructuration, soit la détection et la correction des erreurs existantes liées au modèle conceptuel de l'ontologie et ne tentent pas d'identifier d'autres connaissances aussi pertinentes pour le domaine modélisé et qui sont manquantes dans l'ontologie actuelle. Pour répondre à cette problématique, nous faisons l'hypothèse que L'Analyse relationnelle de concepts (ARC) pourrait constituer un cadre formel très approprié pour la restructuration d'une ontologie, notamment pour l'identification des connaissances manquantes.

Le but de cette thèse est de proposer une approche de restructuration des ontologies par l'ARC. Cette approche devrait assurer trois étapes principales : (1) le codage de l'ontologie initiale dans le format d'entrée de l'ARC ; (2) l'application des méthodes de l'ARC pour l'analyse des données de l'ontologie et leur organisation sous formes de treillis de concepts formels ; et (3) la génération de l'ontologie restructurée à partir des treillis construits.

Notre principale contribution serait de gérer la complexité de ces treillis de concepts par le filtrage des éléments pertinents qui vont constituer les éléments de l'ontologie restructurée. Notre objectif est de proposer une nouvelle méthode spécialement adaptée pour faire un filtrage informé par l'ontologie de départ.

Mots clés : Restructuration des ontologies, Analyse formelle de concepts, Analyse relationnelle de concepts, filtrage des treillis de concepts, mesures d'évaluation de pertinence.

CHAPITRE I

INTRODUCTION

La notion d'ontologie en informatique est née du besoin de représenter les connaissances que l'on a sur le monde de façon à ce qu'elle soit utilisable par des machines pour des fins d'inférences. Le champ d'application des ontologies ne cesse de s'élargir et couvre plusieurs domaines (ex. gestion des connaissances, aide à la décision, ingénierie des systèmes intelligents, enseignement assisté par ordinateur, etc.). Au fur et à mesure qu'il y ait des projets qui se basent sur l'utilisation des ontologies, une véritable ingénierie se constitue autour de ces ontologies en termes de méthodologies et d'outils de développement. Cette ingénierie a pour but non seulement la construction d'ontologies, mais également leur gestion tout au long d'un cycle de vie (Fürst, 2002).

L'Ingénierie ontologique (IO) joue un rôle important dans l'avancement de la recherche dans plusieurs domaines d'applications, notamment pour la modélisation des connaissances théoriques et pratiques des différents domaines et la création d'un consensus au sujet de ces connaissances. Cependant, l'utilisation des ontologies construites est influencée par leur qualité en terme de pertinence par rapport aux domaines qu'elles modélisent. Malheureusement, cette qualité n'est pas toujours assurée du fait qu'elle peut être affectée par plusieurs facteurs liés aux différentes opérations effectuées lors du processus de développement, à savoir la

construction, l'évolution, la maintenance, etc. Toutes ces opérations peuvent générer des problèmes, des imperfections et des anomalies d'ordre structurel et/ou sémantique qui peuvent mener éventuellement à la détérioration de la qualité des ontologies construites. De ce fait, il est important de se doter d'une opération de restructuration pouvant mettre en évidence ces anomalies et proposer des solutions pour leur correction. Cette opération ayant pour but d'amliorer la qualité structurelle et sémantique d'une ontologie aura un impact sur l'optimisation des inférences et des traitements effectués sur cette ontologie.

C'est dans ce contexte que s'inscrit notre travail de recherche qui vise la proposition d'une nouvelle méthode pour la restructuration des ontologies.

1.1 Problématique

La qualité de l'ontologie peut être affectée par plusieurs facteurs, notamment la présence de redondances, l'absence de certaines informations pertinentes, des problèmes de classification, des problèmes de description, etc.

Ces facteurs sont liés aux différentes phases du processus de développement de l'ontologie. En premier lieu, lors de la construction d'une ontologie, notamment à partir de textes, les outils d'extraction (semi-)automatique présentent une limite liée au fait qu'ils se basent sur l'hypothèse que tous les éléments utiles à l'élaboration d'une ontologie se trouvent nécessairement dans le corpus. Cependant, des abstractions pertinentes peuvent ne pas être mentionnées directement dans le texte. Également, durant la construction d'une ontologie avec des outils de modélisation, de nombreux éléments peuvent échapper à l'attention du développeur menant ainsi à des erreurs de conception. En deuxième lieu, une ontologie lors de son évolution peut subir des changements simples (modification, ajout, suppression, etc.) et/ou complexes (fusion, division, etc.) des éléments ontologiques.

Tous ces changements peuvent éventuellement mener à des incohérences au niveau de l'ontologie, telles que : la redondance des éléments de modélisation, l'absence d'éléments pertinents, la génération d'éléments indépendants qu'il convient de relier, etc. Ces problèmes peuvent également surgir, en troisième lieu, lors de la construction d'une ontologie par assemblage de plusieurs modules ou composantes ontologiques couvrant des parties du domaine à modéliser.

Les types d'erreurs, d'imperfections et d'anomalies qui peuvent être identifiés dans une ontologie consistent principalement en (Baumeister et Seipel, 2005; Gómez-Pérez, 2001) : (1) *Inconsistance*, générée par des erreurs de circularité, de partitions et/ou des erreurs sémantiques ; (2) *Incomplétude*, dans le cas où des connaissances pertinentes sont inexistantes dans l'ontologie ; (3) *Redondance* de certains éléments ontologiques (classes/instances ou bien relations taxonomiques) ; et (4) *Déficience*, qui représente d'autres types d'anomalies, citons : *classe ou propriété inutile*, *chaine d'héritage*, *classe disjointe isolée*, *co-domaine de propriété trop spécifique*, *redondance dans les affectation des propriétés*, etc. Ces anomalies n'impliquent pas nécessairement des erreurs au niveau du raisonnement sur l'ontologie, mais correspondent à des imperfections qu'il faudrait modifier pour améliorer le « design » de l'ontologie et par conséquent faciliter sa compréhension et sa maintenabilité.

Ainsi, comme la plupart des projets d'ingénierie des systèmes d'information (modèles conceptuels et code sources), un modèle ontologique est également sujet à des erreurs et des anomalies. Par analogie au domaine de l'ingénierie des logiciels, des travaux existants proposent des méthodes de restructuration pour la correction de ces anomalies.

L'état actuel de la recherche dans le domaine de la restructuration des ontologies nous a permis de constater l'intérêt de plus en plus croissant que la communauté de

l'IO porte à la tâche de restructuration, et comment cette tâche joue un rôle de plus en plus imporant dans le processus de développement des ontologies (Fernández-López *et al.*, 1997; Gómez-Pérez, 1998; Suarez-Figueroa *et al.*, 2012).

De notre analyse des approches de restructuration existantes, il en ressort que :

1. Les approches ont tenté de proposer des méthodes de restructuration d'une ontologie dans une certaine perspective ou dans un contexte donné, et ne sont donc pas généralisables. Cela rend difficile leur comparaison ;
2. Il n'existe pas de méthodologie bien établie qui couvre tout le processus de restructuration et qui permet de faire un remaniement global de la structure de l'ontologie. La plupart des approches effectue des remaniements locaux visant à corriger les anomalies détectées ;
3. Les travaux existants s'attachent plus à un seul niveau de la restructuration, à savoir la correction des erreurs de conception liées à la structure de l'ontologie. Le deuxième niveau, celui concernant la découverte des connaissances manquantes n'est pas bien pris en compte ;
4. Pas de bonne compréhension de ce que le processus de restructuration doit couvrir en tant que tâches de détection de problèmes et tâches de correction et donc il manque un cadre formel pour la résolution conjointe de ces deux types de tâches ;
5. Dans la majorité des approches, les opérations de restructuration sont faites manuellement par les développeurs.

De ces constats par rapports aux limites des approches existantes dérivent les questions de recherche suivantes :

Q1 : Comment détecter et corriger les anomalies qui peuvent exister dans une ontologie ? Cette question a été traitée par la majorité des travaux développés. Le schéma prédominant consiste à proposer des définitions d'un ensemble sou-

vent peu exhaustif d'anomalies, utiliser ces définitions pour la détection de leurs manifestations au sein de l'ontologie et ensuite appliquer des méthodes, le plus souvent manuelles, pour la correction des erreurs détectées. Dans ce travail, notre vision est un peu différente dans le sens où ces travaux effectuent des remaniements locaux au niveau de l'ontologie suite à la détection d'une anomalie alors que nous désirons répondre à cette question par une réorganisation exhaustive et automatique de l'ontologie.

Q2 : Peut-on garantir une réorganisation complète de la structure d'une ontologie sans nuire à sa sémantique ? Cette question n'a pas été traitée par les approches de restructurations citées dans la littérature. Le but de notre travail est d'apporter des éléments de réponses à cette question par la proposition d'un cadre global de la restructuration. Un cadre qui permettra de faire une réorganisation complète de la structure de l'ontologie sans que les modifications effectuées sur la structure ne changent la sémantique véhiculée par la structure de l'ontologie initiale. Ce qui signifie que dans le cas d'une ontologie OWL, par exemple, les triplets RDF énoncés sur les concepts qui étaient valides avant la restructuration vont être compatibles avec la nouvelle ontologie restructurée.

Q3 : Comment évaluer la performance d'une méthode de restructuration ? Il est important de nous interroger sur la qualité des résultats de notre approche de restructuration. Dans notre travail, nous ne comptons pas évaluer la pertinence de la restructuration mais plutôt valider la performance de notre méthode de restructuration.

1.2 Hypothèses de recherche

Afin de rendre nos questionnements plus précis, nous avons pris un certain nombre d'hypothèses qui reflètent notre conviction que : à l'instar de *l'Analyse formelle*

de concepts (AFC) qui a été utilisée avec succès pour la restructuration des modèles en Génie logiciel (GL), son extension vers les Logiques de description (LD) *l'Analyse relationnelle de concepts (ARC)* devrait nous aider dans notre recherche. Ainsi, nos hypothèses se présentent comme suit :

Hypothèse principale : L'Analyse relationnelle de concepts pourrait constituer un cadre formel approprié pour la restructuration d'une ontologie, notamment pour l'identification des connaissances manquantes.

Notre idée est de proposer une approche de restructuration des ontologies qui soit soutenue par un cadre formel capable de garantir, d'une part, le respect des principes ontologiques, à savoir les notions de concepts, propriétés, rôles, relations hiérarchiques, etc. et, d'autre part, de faire une restructuration complète de ces éléments. Pour cette raison, nous pensons que *l'Analyse relationnelle de concepts* pourrait constituer ce cadre. Cette hypothèse se base sur le fait que les ontologies et l'ARC modélisent des concepts, mais dans des buts différents. Le but d'une ontologie est la modélisation des connaissances d'un domaine et la création d'un consensus au sujet de ces connaissances. En revanche, le but de l'ARC est l'analyse et la structuration des données d'un domaine. Un concept en ARC est défini par une paire d'ensembles : un ensemble d'objets (extension) qui représentent les instances de ce concept dans le domaine, et un ensemble d'attributs (intension) qui représentent les descripteurs de ce concept. Par contre, la définition d'un concept ontologique se focalise sur la partie intentionnelle (Formica, 2006).

En outre, l'utilisation de l'AFC/ARC a été explorée dans différents domaines de recherche et a donné de bons résultats. Notons ceux qui nous intéressent : (1) l'ingénierie ontologique, notamment la construction et la fusion des ontologies (Stumme et Maedche, 2001; Cimiano *et al.*, 2005; Nanda *et al.*, 2006; Bendaoud *et al.*, 2008; Rouane-Hacene *et al.*, 2008) ; et (2) la réingénierie des modèles (Godin et

Mili, 1993; Godin *et al.*, 1998; Snelting, 2000; Godin et Valtchev, 2005; Rouane-Hacene *et al.*, 2007).

Sous-hypothèses :

S-Hyp1 : Les éléments ontologiques pourraient être codés en une structure manipulable par l'ARC. Les méthodes de l'ARC reçoivent en entrée les données à analyser sous forme de tables binaires codant un ensemble d'individus et leurs descripteurs associés. Étant donné qu'une ontologie est une structure hiérarchique de concepts, nous supposons que les éléments ontologiques (principalement les concepts) peuvent facilement être représentés par des tables binaires codant un ensemble de concepts décrits par un ensemble de propriétés. Le même principe sera appliqué pour le codage des autres éléments, notamment les relations sémantiques entre les concepts.

S-Hyp2 : L'ARC pourrait assurer une réorganisation complète de la structure d'une ontologie sans perte d'informations, donc améliorer sa qualité tout en préservant (au moins) sa sémantique. Les méthodes de l'ARC se basent sur les données en entrée pour faire des regroupements d'individus ayant des propriétés en commun et les organiser sous forme de structures hiérarchiques appelées « treillis ». Grâce aux propriétés intrinsèques des treillis telles que la factorisation maximale et l'exhaustivité, l'ARC nous fournit un environnement naturel de représentation, d'organisation et de structuration des données (Rouane-Hacene *et al.*, 2007). La factorisation maximale assure une absence totale des redondances, et l'exhaustivité permet au treillis d'inférer toutes les abstractions possibles à partir des données existantes. Ainsi, dans notre contexte des ontologies, les connaissances déjà existantes seront conservées et de nouvelles abstractions seront extraites et intégrées à l'ontologie initiale. Pour cela, nous pouvons faire l'hypothèse que l'ARC pourrait assurer une réorganisation complète de la structure d'une ontologie sans

perte d'informations sur la sémantique véhiculée par cette structure.

S-Hyp3 : Les résultats issus de l'ARC pourraient être traduits vers une nouvelle structure ontologique. Les résultats issus des méthodes de l'ARC sont des treillis de concepts formels. Nous faisons l'hypothèse que cette structure peut être traduite en une ontologie en s'appuyant sur le fait que théoriquement un concept formel correspond à un concept ontologique (Bain, 2003). D'autres travaux (Cimiano *et al.*, 2004; Formica, 2006) font correspondre un concept ontologique à l'ensemble des attributs d'un concept formel.

S-Hyp4 : Un filtrage efficace des treillis issus de l'ARC pourrait améliorer la qualité de la restructuration. Bien que l'ARC a l'avantage d'être exacte et complète, elle possède l'inconvénient de générer des treillis très complexes avec beaucoup de concepts superflus qu'il faudrait éliminer. Nous faisons l'hypothèse qu'il est possible de détecter les concepts superflus à l'aide de critères généraux qui exploitent la structure du treillis, l'ontologie initiale et possiblement des sources de connaissances universellement accessibles.

1.3 Objectifs de la recherche

L'objectif global de cette recherche consiste à proposer une nouvelle approche de restructuration basée sur l'ARC qui vise l'amélioration de la qualité structurelle et sémantique d'une ontologie.

Le premier objectif spécifique vise à développer un cadre structurel et formel couvrant tout le processus de restructuration et permettant de faire une réorganisation complète d'une ontologie.

Le deuxième objectif spécifique vise à développer une méthode pour le filtrage des treillis de concepts issus de l'ARC. Un filtrage qui soit basé essentiellement

sur l'évaluation de la pertinence relative des concepts formels qui vont constituer l'ontologie restructurée.

1.4 Contributions

Ce travail apporte une contribution à la restructuration des modèles ontologiques par la proposition d'une nouvelle méthode basée sur l'ARC. Dans ce cadre, nos contributions sont résumées comme suit :

Une approche globale pour la restructuration d'ontologies

- L'approche couvre tout le processus de restructuration permettant la prise en compte conjointe des deux aspects suivants. Tout d'abord, elle permet l'identification et la correction des anomalies existantes dans l'ontologie. Ensuite, la découverte et l'intégration des connaissances manquantes.
- L'approche est complètement automatique. Elle se base sur un cadre formel avec des méthodes universellement applicables qui effectue une réorganisation complète des éléments ontologiques de sorte que les anomalies visées soient détectées et corrigées automatiquement et de manière exhaustive.
- L'approche fournit, comme pour les classes, une structure hiérarchique des « propriétés objet ».
- La méthode de codage proposée, basée sur un méta-modèle simplifié d'ontologies, suit un processus général et extensible. Ce processus de codage permettra donc de représenter et coder les principaux éléments du méta-modèle OWL complet en une FCR (structure d'entrée de l'ARC).
- L'approche, bien qu'elle soit proposée pour la restructuration des ontologies OWL, peut être appliquée pour d'autres types de langages ontologiques (ex. DAML+OIL). Une adaptation sera nécessaire uniquement au niveau : (1) des règles de codage de l'ontologie représentée dans le langage considéré; et (2)

des règles de traduction des concepts formels filtrés vers le langage ontologique cible.

- L'approche est généralisable, elle ne dépend pas d'un contexte ou d'une ontologie de domaine spécifique.
- L'approche offre un cadre de formalisation pour la génération d'une ontologie OWL à partir d'une représentation conceptuelle (concepts, propriétés, relations, etc.) d'un domaine. Ceci est possible avec une adaptation de la méthode du codage.

Une méthode de filtrage des treillis de concepts

- La méthode développée représente une solution à la problématique du filtrage des treillis de concepts volumineux dans un contexte de restructuration.
- Des mesures pour l'évaluation de la pertinence relative de concepts formels dans un contexte ontologique.
- Une combinaison de mesures qui améliore les performances du processus. Le travail réalisé a révélé qu'un filtrage basé sur une combinaison de mesures apporte une amélioration significative à l'outil de restructuration des ontologies en comparaison avec le filtrage basé sur un algorithme naïf ainsi qu'avec le filtrage basé respectivement sur chacune des mesures retenues.
- Les précédents constituent des moyens pour gérer la complexité des treillis. Les méthodes développées dans cette thèse pourraient être réutilisées dans d'autres contextes d'utilisation de l'AFC et de l'ARC, là où il serait nécessaire de filtrer des treillis de concepts et notamment d'évaluer la pertinence relative des concepts formels au sein de ces structures, citons les autres opérations de l'ingénierie ontologique (construction/apprentissage, fusion, modularisation, etc.) ainsi que le domaine de la réingénierie de modèles.

1.5 Organisation de la thèse

Le chapitre 1 (Introduction) a présenté le contexte général dans lequel s'inscrit cette recherche, posé la problématique et les questions de recherche qui nous ont préoccupé, donné les hypothèses que nous avons prises afin de rendre nos questionnements plus précis, défini les objectifs de recherche, et finalement présenté les principales contributions de cette thèse.

Le chapitre 2 (État de l'art) est scindé en deux grandes sections. La première propose un état de l'art sur le domaine de l'IO. Nous commençons par présenter les concepts de base liés à l'ingénierie ontologique (définitions, principes, méthodes et outils, etc.). Ensuite, nous définissons les aspects problématiques (anomalies) qui peuvent exister dans un modèle ontologique. La deuxième section offre une revue de littérature des approches de restructuration des ontologies. En premier, nous définissons la problématique de la restructuration d'un modèle ontologique. Ensuite, nous résumons les principaux travaux qui ont déjà adressé cette problématique. Enfin, nous présentons une analyse critique des approches existantes tout en précisant les problèmes liés aux ontologies non encore traités par ces approches.

Le chapitre 3 (Cadre de restructuration des ontologies par l'ARC) présente notre approche pour la restructuration des ontologies. Nous entamons le chapitre par la présentation de notre cadre théorique qui est l'Analyse relationnelle de concepts (ARC). Ensuite, nous décrivons le processus général de la restructuration par l'ARC en détaillant ses différentes phases. Enfin, nous précisons comment notre cadre peut traiter un certain nombre d'anomalies qui peuvent exister dans un modèle ontologique.

Le chapitre 4 (Méthode pour le filtrage des treillis de concepts) constitue la contribution principale de cette thèse. Il est entamé par la formulation de la problématique du filtrage des treillis. Ensuite, développe un certain nombre de mesures que

nous avons retenues dans le but de trouver une solution à cette problématique dans notre contexte de restructuration.

Le chapitre 5 (Évaluation de l'approche) est consacré à l'évaluation de l'approche proposée. En premier lieu, les outils et les plateformes utilisés pour l'implémentation des modules développés sont présentés. Deuxièmement, le protocole suivi pour la validation de la méthode de filtrage et de l'approche globale est décrit. Ensuite, les détails des expérimentations menées sont donnés. Finalement, les résultats des expérimentations sont analysés et discutés.

Le chapitre 6 (Conclusion) donne un résumé des réalisations et présente les principales contributions de la présente thèse suivis des limites du travail réalisé et des perspectives envisagées dans ce cadre de recherche.

CHAPITRE II

ÉTAT DE L'ART

2.1 Ingénierie ontologique

Le terme « Ingénierie ontologique (IO) » a été introduit dans (Mizoguchi et Ikeda, 1998) pour désigner un nouveau champ de recherche en Intelligence artificielle (IA) qui vise le développement de systèmes informatiques orientés contenus. Ce champs est née du besoin de résoudre des problèmes liés au partage et réutilisation des connaissances, à la représentation des connaissances nécessitant un certain consensus, à l'intégration de sources de données hétérogènes, etc. qui nécessite non seulement des théories ou des méthodes de raisonnement, mais également un traitement adéquat de la sémantique des représentations.

Cette section présente l'état de l'art sur l'IO. Elle est organisée en cinq sous-sections. La première présente la notion d'ontologie en donnant sa définition (en philosophie et en informatique), ses composantes et sa typologie. La deuxième définit le cycle de vie d'une ontologie. Les trois sous-sections suivantes décrivent respectivement les principes qui guident le processus de construction des ontologies, les méthodologies et les outils qui soutiennent ce processus. Enfin, la dernière sous-section donne la liste de problèmes, d'imperfections et d'anomalies qui peuvent être identifiés au niveau d'une ontologie.

2.1.1 Notion d'ontologie

L'ontologie est un terme philosophique qui désigne l'étude de ce qui existe, c'est-à-dire les connaissances que l'on a sur le monde (Welty et Guarino, 2001).

Les ontologies sont apparues en IA avec l'émergence de l'Ingénierie des connaissances (IC) comme réponses aux problématiques de représentation et de manipulation des connaissances au sein des systèmes informatiques (Fürst, 2002). L'IC est le domaine de développement d'expertise en conception des Système à base de connaissances (SBC). Un SBC permet le stockage, la consultation et la manipulation des connaissances, et leur partage entre plusieurs systèmes informatiques ainsi que le raisonnement automatique sur les connaissances stockées. Dés l'or, des progrès considérables ont été constatés dans le développement des bases conceptuelles qui permettent la réutilisation et le partage des connaissances. Une des avancées est le fait de fonder ces bases de connaissances sur la spécification d'une ontologie. Cette approche permet d'établir des représentations à travers lesquelles les machines peuvent manipuler la sémantique des informations, et assurer ainsi leur interprétation (Fürst, 2002).

Au cours de cette dernière vingtaine d'années, plusieurs définitions ont été attribuées à la notion d'ontologie. La définition considérée comme historiquement la première en IA est celle de (Gruber *et al.*, 1993) : « une ontologie est une spécification explicite d'une conceptualisation ». À ceci, (Borst, 1997) a ajouté : « une ontologie est définie comme étant une spécification explicite et formelle d'une conceptualisation partagée ». Cette définition a été expliquée par (Studer *et al.*, 1998) comme suit : « une spécification **explicite** et **formelle** d'une **conceptualisation partagée** », telle que :

explicite : Tous les concepts, les relations, les propriétés, les contraintes, les fonctions, et les axiomes sont définis explicitement.

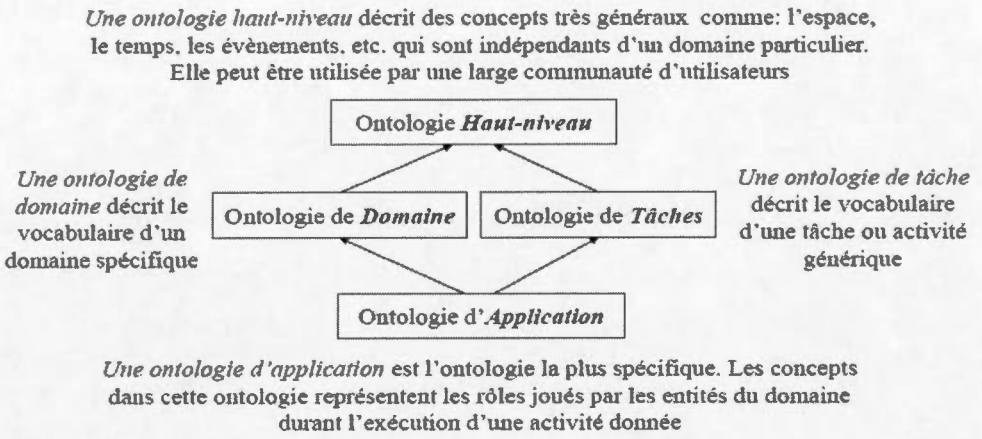


Figure 2.1 Typologie des ontologies selon l'objet de conceptualisation (Guarino, 1998).

formelle : L'ontologie peut être traduite dans un langage interprétable par la machine.

conceptualisation : Un modèle abstrait qui correspond à l'identification des concepts appropriés à un phénomène dans le monde.

partagée : Toutes les connaissances détenues dans l'ontologie sont partagées par un groupe ou une communauté.

Plusieurs typologies des ontologies ont été proposées dans la littérature. Ces typologies se diffèrent par leur critère de classification, à savoir : *niveau de détail*, *niveau de complétude*, *niveau de formalisme de représentation*, et enfin *objet de conceptualisation* (Mizoguchi et Bourdeau, 2004).

La typologie la plus connue est celle basée sur *l'objet de conceptualisation* (Guarino, 1998) illustrée par le schéma de la figure 2.1.

Les connaissances dans une ontologies sont modélisées en utilisant cinq types d'éléments (Gómez-Pérez, 1999) :

- *Concepts :* Un concept définit toute connaissance pertinente décrivant une partie de la réalité. Un concept peut être concret ou abstrait, élémentaire ou composé, réel ou fictif. Il peut donc représenter un objet matériel, une idée, une tâche, une fonction, etc.
- *Relations :* Les relations modélisent les types d'interaction entre les concepts d'un domaine et permettent de raffiner la sémantique de ces concepts. Ces relations sont de type : généralisation-spécialisation, composition, dépendance, disjonction, etc.
- *Fonctions :* Les fonctions sont des cas particuliers de relations. Dans le cas d'une fonction, le n-ième élément de la relation est défini en fonction des (n-1) éléments précédents.
- *Axiomes :* Les axiomes représentent des assertions, considérées comme toujours vraies, utilisées pour modéliser des affirmations évidentes sur le domaine.
- *Instances :* Les instances représentent les objets manipulés à travers les concepts.

2.1.2 Cycle de vie des ontologies

Étant donné que les ontologies sont considérées comme des composantes logicielles destinées à être utilisées dans des systèmes informatiques, elles possèdent un cycle de vie qui nécessite d'être spécifié (Fürst, 2002). Les auteurs dans (Dieng *et al.*, 2001) ont proposé un cycle de vie inspiré du génie logiciel comme illustré à la figure 2.2. Il comprend six principales étapes : (1) *détection des besoins* ; (2) *construction* ; (3) *diffusion* ; (4) *utilisation* ; (5) *évaluation* ; et (6) *évolution.* Le suivi et la gestion des itérations sur ces étapes sont assurés par une activité de contrôle et de planification.

Détection des besoins : Cette étape consiste à délimiter le domaine de connaissances à modéliser, préciser l'objectif opérationnel et identifier les utilisateurs de l'ontologie.

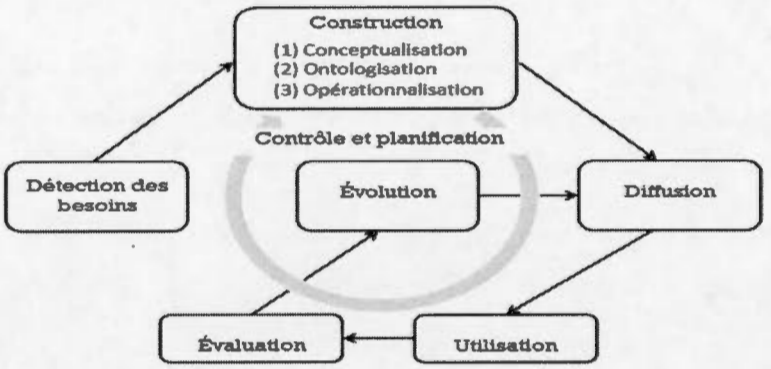


Figure 2.2 Le cycle de vie d'ontologies (Dieng *et al.*, 2001).

Construction : Cette étape est généralement découpée en : (1) *conceptualisation* ; (2) *ontologisation* ; et (3) *opérationnalisation. La conceptualisation* consiste à identifier les connaissances du domaine. Ensuite, préciser la nature conceptuelle de ces connaissances : concepts, relations, propriétés des concepts et des relations, règles, contraintes, etc. *L'ontologisation* est la construction proprement dite de l'ontologie qui doit respecter les principes généraux et les critères de l'ingénierie ontologique (voir section 2.1.3). Dans cette étape, les connaissances définies a priori en langage naturel seront traduites dans un certain formalisme. Le but de l'étape *d'opérationnalisation* est de rendre l'ontologie opérationnelle, c'est-à-dire permettre aux machines de manipuler les connaissances représentées dans l'ontologie, et effectuer des raisonnements sur ces connaissances.

Diffusion : Cette étape s'intéresse au déploiement et à la mise en place de l'ontologie construite.

Utilisation : C'est l'utilisation proprement dite de l'ontologie par les utilisateurs : annotation des ressources, résolution de requêtes, raisonnement, etc.

Évaluation : Cette étape s'intéresse à l'évaluation de la qualité et de l'intérêt de l'ontologie dans son contexte d'usage. Le but est de vérifier si l'ontologie

construite répond aux besoins opérationnels de ses utilisateurs.

Évolution : Cette étape implique la modification de l'ontologie suite à des changements dans le domaine de connaissances, dans la conceptualisation, ou dans les exigences des utilisateurs.

Les sous-sections suivantes sont consacrées aux principes, méthodologies et outils mis en œuvre lors de la phase de construction.

2.1.3 Principes de construction d'ontologies

Dans (Gruber *et al.*, 1993), l'auteur a proposé un ensemble de critères pour guider le processus de développement des ontologies. Ces critères sont résumés dans ce qui suit :

Clarté : Les définitions des termes d'une ontologie doivent être objectives et doivent communiquer la signification réelle (attendue) des termes définis. Toutes les définitions doivent également être documentées en langage naturel.

Cohérence : Une ontologie doit être cohérente, c'est-à-dire les inférences faites via les axiomes définis au niveau de cette ontologie doivent être conformes aux définitions des concepts.

Extensibilité : Un modèle ontologique doit être extensible de façon monotone, c'est-à-dire de nouveaux termes généraux et/ou spécialisés peuvent être définis dans l'ontologie d'une façon qui ne nécessite pas la révision des définitions déjà existantes.

Biais de codage minimal : Un biais de codage résulte lorsque les choix de représentation des connaissances dépendent d'une notation ou d'une implémentation. Ce biais de codage doit être minimal, car des agents partageant des connaissances peuvent être implémentés dans différents systèmes de re-

présentation.

Engagements ontologiques minimaux : Une ontologie doit exiger des engagements ontologiques minimaux suffisants pour supporter les activités espérées du partage des connaissances. Elle doit spécifier le moins possible ses concepts permettant aux parties engagées dans cette ontologie de spécialiser et d'instancier les concepts de l'ontologie selon leurs besoins.

Les auteurs dans (Azpírez *et al.*, 1998) ont également présenté un ensemble de principes à respecter dans la conception d'une ontologie :

Modularité : Une ontologie doit être modulaire. Il s'agit de minimiser les couplages entre les modules permettant ainsi plus de flexibilité et un usage plus varié.

Diversification de chaque hiérarchie : Ce principe vise l'augmentation de la puissance fournie par les mécanismes de l'héritage multiple. Par une représentation suffisante des connaissances et une utilisation de différents critères de classification, il serait plus facile d'ajouter de nouveaux concepts puisqu'ils peuvent être facilement spécifiés à partir des concepts et des critères de classification préexistants.

Distance sémantique minimale entre les concepts enfants : Au niveau ontologie, les concepts similaires doivent être regroupés et représentés comme des sous-classes d'une même classe tandis que les concepts moins similaires sont représentés plus loin dans la hiérarchie.

Normalisation des noms : Les noms des éléments doivent être (de préférence) normalisés, par exemple il s'agit de spécifier que le nom d'une relation doit être formé en concaténant : le nom du concept source de cette relation, son nom et le nom de son concept destination.

Selon (Gruber *et al.*, 1993), comme la plupart des projets de conception, la conception d'une ontologie exige un compromis entre ces différents critères.

2.1.4 Méthodologies de construction d'ontologies

La construction d'ontologies est l'application d'un ensemble de méthodes et techniques pour la construction d'ontologies à partir des sources d'informations et des connaissances disponibles qui sont généralement distribuées et hétérogènes (Katsiouli, 2007). Ce processus permet de réduire le temps et l'effort nécessaires pour le développement manuel des ontologies.

Les méthodologies recensées dans la littérature pour la construction d'ontologie peuvent être divisées en cinq groupes : (1) *construction à partir de zéro* ; (2) *construction par intégration ou fusion d'ontologies partielles* ; (3) *construction par extension d'ontologie existante* ; (4) *construction par réingénierie* ; (5) *construction collaborative*. Nous avons tenté de résumer les principales méthodologies et approches proposées et développées dans le cadre des trois premières classifications.

Construction d'ontologies à partir de zéro

Un ensemble de méthodologies générales de la littérature (Fernández-López, 1999; Gómez-Pérez, 1999; Fürst, 2002) sont décrites dans ce qui suit.

La méthodologie de (Uschold et King, 1995) est basée sur leur expérience de construction de « Enterprise Ontology » qui inclut un ensemble d'ontologies pour la modélisation des entreprises. Elle comporte les étapes suivantes : (1) identification du but et de la portée de l'ontologie ; (2) construction de l'ontologie par : l'acquisition des connaissances, le codage de ces connaissances, et leur intégration dans des ontologies existantes ; (3) évaluation de l'ontologie ; (4) documentation ; et (5) définition des directives pour chaque phase.

La méthodologie de (Grüninger et Fox, 1995) est également inspirée de l'expérience des auteurs dans le cadre du projet TOVE (TOronto Virtual Enterprise)

pour la construction d'ontologies dans le domaine de la gestion des entreprises. La méthodologie se base sur l'utilisation des questions de compétences, c'est-à-dire les questions auxquelles le système utilisant l'ontologie est censé pouvoir répondre. Les questions de compétences sont tout d'abord exprimées de façon informelle. Elles spécifient les problèmes ainsi que leurs bonnes solutions et vont servir de base à la phase de conceptualisation pour l'extraction des connaissances du domaine que doit couvrir l'ontologie. Une fois ces connaissances formalisées, le même formalisme sera utilisé pour spécifier les questions de compétences de façon formelle qui peuvent être utilisées par la suite pour la validation de l'ontologie construite.

METHONTOLOGY (Fernández-López *et al.*, 1997; Gómez-Pérez, 1998) est une méthodologie qui couvre tout le cycle de vie d'une ontologie. Elle inclut : (1) l'identification du processus de développement de l'ontologie ; (2) le cycle de vie ; et (3) la méthode. Le processus de développement permet d'identifier les activités à effectuer lors de la construction d'une ontologie, soit des activités de gestion de projets (planification, contrôle, assurance qualité), des activités de développement (spécification, conceptualisation, formalisation) et des activités de support (évaluation, documentation, gestion de la configuration). Le cycle de vie, basé sur des prototypes évolutifs, permet d'identifier les étapes par lesquelles passe l'ontologie durant son existence. La méthode spécifie les étapes à suivre pour réaliser chaque activité, les techniques utilisées, les produits en sortie, et les méthodes permettant leur évaluation.

La méthodologie de (Staab *et al.*, 2001) consiste en l'accomplissement des étapes suivantes : (1) l'identification du problème à résoudre, l'étude d'opportunité et le choix de l'ontologie cible ; (2) la spécification des exigences, l'analyse des sources d'information et la création d'une ontologie initiale ; (3) le raffinement de l'ontologie initiale par l'ajout de nouveaux concepts, relations et axiomes ; (4) l'évaluation ;

et (5) la maintenance.

La méthodologie *OntoSpec* de (Kassel, 2002) porte sur la phase d'ontologisation. Elle offre une aide à la structuration des hiérarchies de concepts et de relations.

À la différence des autres approches qui fournissent un guide méthodologique pour l'IO, la méthodologie *NeOn* (Suarez-Figueroa *et al.*, 2012) ne décrit pas un cadre rigide, mais plutôt suggère une variété de moyens pour développer des ontologies. Le cadre de la méthodologie NeOn fourni : (1) un glossaire de processus (ex. alignement, réingénierie, validation, etc.) et d'activités (ex. fusion, évaluation, restructuration, comparaison, etc.) impliqués dans le développement des ontologies ; (2) un ensemble de neuf scénarios pour la construction d'ontologies et réseaux d'ontologies mettant l'accent sur la réutilisation et la réingénierie des ressources de connaissances (ontologiques et non ontologiques). Chaque scénario est décomposé en différents processus et activités parmi ceux inclus dans le glossaire ; (3) deux modèles de cycle de vie d'ontologies qui spécifient comment organiser les processus et les activités dans des phases.
La méthodologie a été appliquée non seulement au développement des réseaux d'ontologies associés aux cas d'utilisation du projet NeOn, mais également dans d'autres domaines comme l'éducation (Clemente *et al.*, 2011), le tourisme (Lamsfus *et al.*, 2009) et les environnements mobiles (Poveda Villalon *et al.*, 2010).

Il est à préciser que la construction d'ontologie peut être effectuée à partir de différents types de sources de connaissances (Katsiouli, 2007) : sources non structurées, sources semi-structurées ou bien sources de données structurée. Étant donné que les sources informationnelles non structurées (documents textuels en langage naturel comme word, documents pdf et pages web) constituent le format le plus disponible comme entrée au processus de construction (Drumond et Girardi, 2008), la majorité des travaux dans le domaine se sont intéressés à la construction d'on-

tologies à partir de documents textuels.

Plusieurs approches ont été proposées dans cette perspective. Cependant, il n'existe toujours pas de consensus clair sur les tâches exactes à effectuer pour mener le processus d'acquisition (Buitelaar *et al.*, 2005). C'est la raison pour laquelle les auteurs ont proposé un schéma général constitué de six couches résumant les étapes à accomplir pour mener à bien le processus d'apprentissage d'ontologies à partir de corpus textuels, à savoir : *extraction des termes, des synonymes, des concepts, de la taxonomie, des relations,* et enfin *des règles et des axiomes.*

La majorité des approches citées dans la littérature ne suit pas le processus général d'extraction dans son ensemble mais s'attache plus à certaines de ses étapes. Ces approches se basent sur des techniques statistiques, linguistiques ou une combinaison des deux. Des travaux (Biemann, 2005; Zouaq et Nkambou, 2008; Buitelaar *et al.*, 2005; Katsiouli, 2007) ont présenté un état de l'art des techniques et des projets dédiés à l'apprentissage d'ontologie à partir de textes. Nous citons dans ce qui suit certains d'entre eux.

Text-To-Onto (Maedche et Staab, 2000) est un système d'apprentissage d'ontologies à partir de ressources textuelles basé sur l'infrastructure KAON[1]. Il couvre la majorité des étapes du processus d'extraction.

Text2Onto (Cimiano *et al.*, 2005) est un outil conçu pour construire automatiquement des ontologies à partir de ressources textuelles. Il utilise l'architecture GATE[2] pour prétraiter ces ressources. C'est le successeur du système Text-To-Onto dont il diffère par l'introduction de deux nouveaux paradigmes dans le processus d'apprentissage : premièrement, des modèles POMs (Probabilistic Ontology

1. http ://kaon.semanticweb.org/

2. http ://gate.ac.uk/ie/

Models) qui représentent les résultats du système de rattachement des probabilités aux connaissances extraites représentées par des méta-modèles. Deuxièmement, à chaque fois que des modifications sont effectuées dans le corpus, les POMs sont mis à jour automatiquement sans avoir besoin de les recalculer pour toute la collection de documents.

TEXCOMON (TEXt COncept Map ONtology) (Zouaq et Nkambou, 2008) est destiné à générer semi-automatiquement une ontologie de domaine à partir de textes, notamment à partir d'objets d'apprentissage. Cet outil présente deux spécificités : (1) transformation de textes en cartes de concepts ; et (2) formalisation de ces cartes de concepts sous forme d'ontologies du domaine.

TERMINAE (Aussenac-Gilles *et al.*, 2008) est une plateforme d'apprentissage d'ontologies à partir de textes. L'outil TERMINAE est constitué de composants qui exécutent des tâches spécifiques pour construire une ébauche d'ontologie dans une représentation formelle. Une formalisation plus précise consiste à exporter cette ontologie vers un éditeur comme Protégé-OWL. Pour faciliter les échanges des données et des résultats entre les composants, l'outil se base sur un ensemble de fichiers XML.

Plusieurs d'autres travaux se basent sur l'Analyse formelle de concepts (AFC) pour l'extraction d'ontologies à partir de textes (Stumme et Maedche, 2001; Cimiano *et al.*, 2005; Nanda *et al.*, 2006; Bendaoud *et al.*, 2008; Rouane-Hacene *et al.*, 2008). Dans le cadre du processus d'apprentissage, l'AFC permet de construire un treillis de concepts à partir des tables binaires initiales. Ces tables sont construites à partir des individus (par l'extraction des termes qui vont constituer les concepts pertinents du domaine étudié) et de leurs propriétés (par la dérivation des propriétés des termes à partir du texte). Ce treillis permet d'extraire des relations taxonomiques (*is-a*). Les auteurs dans (Rouane-Hacene *et al.*, 2008) ont proposé

une extension à l'AFC pour construire un treillis qui prendra en considération les autres types de relations (transversales) entre concepts. Pour se faire, ils ont proposé une approche semi-automatique basée sur l'Analyse relationnelle de concepts (ARC) dont le principe est de traduire les relations entre objets en « attributs relationnels ».

Construction d'ontologies par fusion (ou assemblage de modules ontologiques)

Les méthodologies de construction d'ontologies par fusion prennent en considération deux (ou plus) ontologies en entrée et retourne une seule ontologie intégrant les ontologies sources. La fusion des ontologies est considérée comme une tâche difficile à effectuer manuellement et peut générer beaucoup d'erreurs (Stumme, 2005). C'est la raison pour laquelle plusieurs systèmes ont étés développés pour assister les ingénieurs dans leur tâche de fusion. Les approches proposées dans (Hovy, 1998; Chalupsky, 2000; McGuinness *et al.*, 2000; Noy et Musen, 2000) s'appuient sur des heuristiques d'alignement syntaxiques et sémantiques inspirées des étapes effectuées manuellement par les développeurs pour la fusion des ontologies. Contrairement à ces approches qui proposent différents types de logiques pour les comparaisons entre les ontologies à fusionner, les auteurs dans (Stumme et Maedche, 2001) proposent une approche « FCA-Merge » pour la fusion d'ontologies basées sur l'AFC offrant une description structurelle globale du processus de fusion. C'est une approche *bottom-up* qui prend en entrée les deux ontologies à fusionner et un ensemble de documents textuels pertinents pour les deux ontologies. Des techniques de traitement du langage naturel sont utilisées pour extraire des instances à partir de ces documents. Ce mécanisme d'extraction d'instances permet de retourner, pour chaque ontologie, un contexte formel indiquant quel concept apparaît dans quel document. Les deux contextes formels sont ensuite fusionnés et un treillis de concepts est construit par les techniques de l'AFC. Enfin, ce treillis est filtré puis transformé en une ontologie fusionnée par un expert.

Extension d'ontologies existantes

C'est le cas où une ontologie existe et on désire faire son extension à partir de documents textuels non structurés, c'est-à-dire extraire des termes à partir du texte et tenter de les assigner aux classes existantes dans la taxonomie. Ce processus d'apprentissage peut être considéré comme une tâche de classification. À cet effet, les techniques d'apprentissage machine (Michalski *et al.*, 2013) basées sur la classification sont généralement utilisées.

Le principe est le suivant : ces algorithmes reçoivent en entrée des données d'entrainement (caractéristiques des données existantes) sur la base desquelles ils créent un modèle. Ensuite, pour chaque terme et en suivant le modèle, ils affectent les nouveaux termes (nouvelles instances) aux classes appropriées déjà existantes ou ajoutent de nouvelles sous-classes.

Un exemple des techniques utilisées est la technique des arbres de décisions (Alfonseca et Manandhar, 2002; Witschel, 2005). D'autres travaux se sont concentrés sur l'élargissement de la structure hiérarchique de Wordnet [3] avec de nouveaux mots (Widdows, 2003; Paşca, 2005).

2.1.5 Outils de construction d'ontologies

Le tableau 2.1 résume les environnements et les outils d'aide à la construction d'ontologies les plus connus. La majorité d'entre eux ont été recensés dans les travaux (Duineveld *et al.*, 2000; Gómez-Pérez, 1999; Fürst, 2002).

3. http ://wordnet.princeton.edu/

Tableau 2.1 Outils d'ingénierie ontologique.

Outil	Brève description
Ontolingua	Ontolingua [a] (Farquhar *et al.*, 1997) est un serveur d'édition d'ontologie au niveau symbolique. Il permet la construction d'une ontologie exprimée directement dans le langage « Ontolingua ». Il propose des outils et des services qui supportent la construction collaborative d'ontologie entre des groupes distribués ainsi que l'intégration d'une ontologie dans celle en cours de construction.
OntoEdit (Ontology Editor)	OntoEdit (Maedche *et al.*, 2000) est un éditeur d'ontologies qui supporte le développement et la maintenance des ontologies. Il permet l'édition des hiérarchies de concepts, relations, axiomes, et propriétés. L'outil intègre un serveur destiné à l'édition collaborative des ontologies et inclut également des modules graphiques pour la visualisation.
ODE (Ontology Design Environment)	ODE (Blázquez *et al.*, 1998) est un outil qui permet la construction d'ontologies au niveau connaissances en utilisant des représentations intermédiaires qui sont indépendantes du langage cible dans lequel l'ontologie sera implémentée. Une fois la conceptualisation est complétée, le code est généré automatiquement en utilisant des générateurs de code ODE (ex. Ontolingua).
Protégé	Protégé [b] est l'outil le plus connu et le plus utilisé pour l'édition d'ontologies. Il est le successeur de ProtégéWin. L'outil est basé sur Java et il offre plusieurs possibilités de modélisation d'ontologies grâce aux éditeurs Protégé-Frames et Protégé-OWL. Il implémente un ensemble riche de structures de modélisation de connaissances et de fonctionnalités qui supportent la création, la visualisation et la maintenance d'ontologies dans différents formats de représentations.
Tadzebao et WebOnto	Tadzebao et WebOnto (Domingue, 1998) sont deux outils complémentaires. Tadzebao permet aux ingénieurs de connaissances de mener des discussions synchrones et asynchrones sur les ontologies, et WebOnto supporte la création, la consultation et l'édition des ontologies sur le web.
NeOn Toolkit	NeOn Toolkit [c] (Haase *et al.*, 2008) est un environnement d'ingénierie ontologique développé dans le cadre du projet NeOn [d]. Il est basé sur la plateforme « Eclipse » et intègre un ensemble de plugins couvrant différentes activités liées à l'ingénierie ontologique telles que : apprentissage d'ontologies (Text2Onto) ; alignement d'ontologies (R2O, FOAM) ; construction d'ontologies par réutilisation d'ontologies existantes (Watson) ; diagnostique et réparation (RaDON) ; et modélisation visuelle (OntoModel). L'outil NeOn Toolkit supporte également l'intégration de plugins distribués incluant par exemple des plugins basés sur des services web pour accéder à des composants distants.

---

a. http ://ksl.stanford.edu/software/ontolingua/

b. http ://protege.stanford.edu/

c. http ://www.neon-toolkit.org/

d. http ://www.neon-project.org/

2.1.6 Aspects problématiques liés à l'ingénierie ontologique

Les méthodes d'IO peuvent produire des erreurs ou des anomalies au niveau des ontologies construites menant à la détérioration de la qualité de ces ontologies en terme de pertinence et de complétude par rapport au domaine modélisé.

La notion d'anomalie a été définie initialement pour les bases de données relationnelles (Baumeister et Seipel, 2005). Ensuite, cette notion a été adoptée ces dernières années par les recherches en génie logiciel qui ont introduit le terme de *« bad smells »* pour décrire des parties de code sources qui n'impliquent pas nécessairement des erreurs au niveau du comportement fonctionnel du logiciel mais qui sont mal conçues et devraient être corrigées afin d'améliorer la compréhension et la maintenance de ce code source (Fowler, 1999). A cet effet, diverses pistes ont été explorées : (1) définir les différents *« bad smells »* qui peuvent exister au niveau d'un code source ; (2) proposer des méthodes pour la détection de ces *« bad smells »* ; et (3) proposer des méthodes de restructuration qui visent l'amélioration de l'aspect *« design »* du code par la correction de ces *« bad smells »*. Par analogie, la recherche sur l'IO a tenté dans un premier temps de définir les anomalies qui peuvent exister dans un modèle ontologique, ensuite proposer des méthodes pour la détection de ces anomalies et enfin proposer des méthodes de restructuration permettant de les corriger.

Les types de problèmes, d'imperfections et d'anomalies qui peuvent être identifiés dans une ontologie sont résumés dans ce qui suit.

Selon (Gómez-Pérez, 2001), les anomalies qui peuvent être identifiées dans une taxonomie peuvent être classées dans trois grandes catégorie : *inconsistance*, *incomplétude* et *redondance*.

Inconsistance : On peut distinguer trois types d'erreurs qui mènent à des inconsistances : *erreurs de circularité*, *erreurs de partitions* et *erreurs sémantiques*.

1. *Erreurs de circularité :* Une erreur de circularité est identifiée dans une ontologie si une classe est une spécialisation ou une généralisation d'elle-même. Étant donnée une classe cl_i, les trois types d'erreurs de circularité sont :
   (a) erreurs de circularité de distance 0 : cl_i *is-a* cl_i (une classe avec elle même),
   (b) erreurs de circularité de distance 1 : cl_i *is-a* cl_j et cl_j *is-a* cl_i,
   (c) erreurs de circularité de distance n : cl_i *is-a* cl_{i+1} *is-a* cl_{i+2}....*is-a* cl_{i+n} *is-a* cl_i.
2. *Erreurs de partitions :* Les partitions dans une taxonomie sont utilisées pour définir des contraintes sur les classifications de classes. *Une partition* définit un ensemble de classes qui sont mutuellement disjointes.
   Une partition de sous-classes de la classe cl n'est pas nécessairement complète, c'est-à-dire il peut exister des instances de la classe cl qui ne sont incluses dans aucune de ses sous-classes (formant la partition). Dans le cas où la partition est complète, on parle d'une *partition exhaustive de sous-classes* de la classe cl, c'est-à-dire la classe cl peut être définie comme une union de toutes ses sous-classes qui forment la partition.
   Une erreur de partition se produit lorsqu'une partition $cl_1, ..., cl_n$ est définie et une ou plusieurs classes $c_1, ..., c_p$ sont des sous-classes communes entre plusieurs classes cl_i.
   Par exemple, ayant défini les deux classes Cat, Dog comme une *partition de sous classes* de Animal, une erreur de ce type va se produire si on définit la classe Siamese comme une sous-classe des deux.
   Par analogie, une instance commune entre deux classes disjointes produit

également une erreur de partition.

3. *Erreurs sémantiques :* Les erreurs sémantiques consistent en l'existence de classifications non correctes sémantiquement, par exemple une classe est définie par erreur comme une sous-classe d'une autre classe sachant qu'il n'y a aucune relation sémantique entre ces deux classes.

   Par exemple, si on définit la classe Student comme une sous-classe de la classe Vehicle. La même chose va se produire avec les instances quand l'instance Marc sera une instance de la classe Vehicle.

Incomplétude : L'incomplétude peut être notée dans les cas suivants :

1. *Classification incomplète de concepts :* Les connaissances peuvent être incomplètes dans le cas où des concepts pertinents sont inexistants dans l'ontologie.

   Par exemple, ce type d'erreurs peut se produire si la classification du concept Vehicle est définie en considérant uniquement les classes Bicycle, Car, Bus et ignorant la classe Truck ; ou bien dans le cas où les trois classes Sports, Sightseeing et Adventure ayant des propriétés communes sont définies au niveau de la taxonomie en oubliant de rajouter leur super-classe Activity qui factorisera ces propriétés.

2. *Informations incomplètes sur les partitions :* Ce type d'erreurs apparait lorsque la définition des partitions est incomplète, c'est-à-dire des connaissances sur la disjonction et l'exhaustivité d'une partition sont incomplètes ou absentes.

   Par exemple, le développeur définit Sports, Sightseeing et Adventure comme des sous-classes de Activity et oublie de préciser que Sports, Sightseeing et Adventure forment une partition de sous-classes (même incomplète) de Activity, donc disjointes.

3. *Incomplétude au niveau des classes :* Dans le cas où des propriétés nécessaires

à la définition d'une classe sont manquantes.

4. *Manque d'informations sur les domaines et les co-domaines :* Le domaine et le co-domaine de chaque argument de chaque fonction ou relation précisent les classes qui sont appropriées pour cet argument. Une erreur survient dans le cas où les domaines et les co-domaines sont imprécis ou trop précis.

Redondance : Ce type d'erreurs se produit dans une ontologie dans les cas suivants :

1. *Définition redondante de classes :* Le cas où deux ou plusieurs classes ont des définitions formelles identiques. La seule différence entre elles c'est le nom.

   Par exemple, si le développeur définit l'entité Enfant par deux classes différentes Kid et Child ayant un même ensemble de propriétés {name, firstname, age}. Même erreur peut exister avec des instances redondantes.

2. *Relations taxonomiques « is-a » redondantes :* Une relation *« is-a »* est considérée comme redondante si elle est répétée dans l'ontologie de façon directe ou indirecte.

   Une répétition directe concerne l'existence d'une même relation *« is-a »* deux ou plusieurs fois entre les mêmes classes *source* et *destination.* Par exemple, si on définit deux fois la relations *« is-a »* entre la classe Man et Person.

   La répétition indirecte se produit lorsque la relation *« is-a »* peut être dérivée à partir d'autres relations *« is-a »*. Par exemple, si on définit la classe Siamese comme une sous-classe de Cat et Cat comme une sous-classe de Animal, et on définit également Siamese comme une sous-classe de Animal.

   Ce même type d'erreurs peut exister avec des relations *« instance de »* redondantes entre une instance et une classe (directes ou indirectes).

Les auteurs dans (Baumeister et Seipel, 2005) ont introduit une autre catégorie d'erreurs, soit *Déficience*, dans laquelle il regroupe les autres types d'anomalies qui ne peuvent être identifiées comme des erreurs d'inconsistance, ni de redondance. Ces anomalies, qui sont généralement générées suite à l'évolution ou l'intégration d'ontologies, ne mènent pas à des erreurs de raisonnement sur l'ontologie mais devraient être éliminées afin d'améliorer la compréhension et la maintenabilité des connaissances contenues dans le modèle ontologique.

Déficience : Les anomalies regroupées dans cette catégorie sont :

1. *Classe ou propriété inutile :* Un élément ontologique (classe ou propriété) est considéré comme inutile s'il n'est jamais utilisé dans la réalité, c'est-à-dire non pertinent pour le domaine d'application de l'ontologie.
2. *Chaine d'héritage :* On appelle chaine d'héritage une partie de la taxonomie constituée d'une longue suite de relations *« is-a »* : cl_1 *is-a* cl_2...*is-a* cl_n dans laquelle chaque classe cl_i n'est contenue dans aucune autre relation *« is-a »* à l'exception de celle dans la chaine (chaque classe contient une seule sous-classe).
3. *Classe disjointe isolée :* Cette anomalie se produit dans le cas où une classe n'est disjointe avec aucune de ses sœurs, mais possède des relations de disjonction avec un ensemble de classes qui sont mutuellement sœurs dans une autre branche de la taxonomie. Cette erreur peut survenir lors d'une modification manuelle de l'ontologie.

   Par exemple, déplacer une classe vers une autre branche de la taxonomie sans adapter en conséquence les relations de disjonction.
4. *Co-domaine trop spécifique de propriété :* Cette erreur peut apparaitre dans le cas où sont attribuées des valeurs trop spécifiques aux co-domaines des propriétés.

   Par exemple, quand le développeur définit manuellement des valeurs pour

la propriété size dans l'ensemble {very big, big, medium, small, very small} alors que dans le domaine d'application de l'ontologie, cela pourrait changer dans le sens que uniquement les valeurs {big, medium, small} seraient plus appropriées.

5. *Redondance dans les affectations des propriétés :* C'est une anomalie qui est due à la répétition d'un même ensemble de propriétés dans la définition de plusieurs classes. Ces propriétés incluent les *« Datatypes properties »* et les *« Object properties »*.

   Par exemple, dans le cas où l'ensemble des propriétés {id, name, pages} sont répétées dans les deux classes définies Magazine et Newspaper. Cette anomalie peut être corrigée par la création d'une nouvelle classe abstraite Publication factorisant l'ensemble des propriétés répétées. Cette erreur peut être vue comme un cas particulier de l'incomplétude.

Toutes ces anomalies peuvent être produites lors du processus de développement d'une ontologie pour diverses raisons. Par exemple pour la construction d'ontologies à partir de textes, les outils d'extraction semi-automatiques se basent sur l'hypothèse que toutes les connaissances utiles pour la représentation du domaine se trouvent dans le texte, alors que d'autres connaissances (abstractions) pertinentes pour le domaine peuvent ne pas être définies explicitement dans le texte, et par conséquent seront manquantes dans l'ontologie construite.
En outre, des redondances au niveau des éléments ontologiques peuvent facilement être générées lors d'une fusion de deux ou plusieurs ontologies.
Un autre exemple concernant l'évolution d'une ontologie. Dans ce cas là, l'ontologie doit subir des changements tels que : ajout, suppression, modification, renommage, fusion, division, etc. des éléments ontologiques, ce qui peut mener éventuellement à des incohérences en termes de redondances et/ou d'inconsistances.

Des travaux existants ont tenté de détecter ces anomalies et ont proposé des

méthodes pour leur correction en passant par un processus de restructuration. Ces méthodes consistent en : la suppression des éléments redondants, la suppression des concepts inutiles, l'identification du bon niveau d'abstraction pour définir un concept, etc. Ces travaux font l'objet de la section suivante.

2.2 Restructuration des ontologies

Le terme restructuration a été introduit par les recherches en ingénierie des logiciels qui indique la modification du code source sans pour autant changer le comportement externe du logiciel (Fowler, 1999). La modification se focalise sur l'amélioration de l'aspect « design » du code plutôt que sur ses fonctionnalités. Par analogie, la restructuration des ontologies doit viser l'amélioration du modèle conceptuel de l'ontologie et par conséquent faciliter sa compréhension et sa maintenabilité (Baumeister et Seipel, 2006).

Dans cette section, nous commençons par donner la définition suivie de la formulation du problème de la restructuration des ontologies. Ensuite, nous faisons une synthèse des travaux actuels dans le domaine et nous finissons par une analyse critique de ces travaux qui discute les limites des approches existantes, notamment les problèmes liés à l'ingénierie ontologique non encore traités par les travaux développés.

2.2.1 Définition

La restructuration d'une ontologie est une opération de remaniement de sa structure actuelle afin d'élaborer une nouvelle structure mieux organisée et plus complète. Elle réfère à : (1) la correction et la réorganisation des connaissances contenues dans le modèle conceptuel initial ; et (2) la découverte des connaissances manquantes et leur intégration dans l'ontologie de départ (Suarez-Figueroa et

Gomez-Perez, 2008).

Selon (Suarez-Figueroa et Gomez-Perez, 2008), le processus de restructuration consiste en deux phases :

Analyse : Le but de cette phase est d'évaluer l'ontologie. Il s'agit de vérifier si l'hiérarchie de l'ontologie ainsi que ses classes, instances, relations et fonctions sont complètes (contiennent toutes les définitions nécessaires pour le domaine), consistantes (pas de contradictions dans l'ontologie), concises (pas de redondances explicites ni implicites), et correctes syntaxiquement.

Synthèse : Cette phase exploite le résultat de la phase d'analyse et cherche à corriger l'ontologie et documenter les changements effectués.

Notons que la restructuration des ontologies est souvent associée aux autres opérations d'ingénierie (la construction, c'est-à-dire à partir de zéro, par extension, par fusion, l'évolution, la maintenance, etc.) au sein du cycle de vie des ontologies dont le but est d'améliorer la qualité structurelle et sémantique des ontologies construites.

2.2.2 Formulation du problème de restructuration

Dans le cadre de notre travail, nous avons tenté de formuler le problème de la restructuration des ontologies comme suit :

Étant donnée une ontologie initiale notée O_{Init}, l'objectif du processus de restructuration est d'obtenir une ontologie O_{Rest} telle que :

- O_{Rest} est le résultat d'application d'un ensemble d'opérations de restructuration $op \in OP_R$ sur O_{Init}, avec :
  OP_R = {supprimer un élément ontologique redondant, introduire un nouveau concept comme abstraction à partir de propriétés communes, corriger une abstraction erronée, ajouter un élément ontologique manquant, etc.} ;

- Un élément ontologique (concept, propriété, relation) existant dans O_{Init} mais non existant dans O_{Rest} est probablement un élément erroné (redondant, inutile, etc.) :

$$\forall e(e \in O_{Init} \wedge e \notin O_{Rest} \rightarrow e \textit{ est probablement erroné})$$

- Un élément ontologique existant dans O_{Rest} mais non existant dans O_{Init} est probablement un nouvel élément pertinent qui manquait dans l'ontologie initiale (abstraction factorisant des propriétés communes, concept oublié, etc.) :

$$\forall e(e \in O_{Rest} \wedge e \notin O_{Init} \rightarrow e \textit{ est probablement pertinent qui manquait dans } O_{Init})$$

- O_{Rest} est de meilleure qualité (plus complète et mieux structurée) que O_{Init} :

$$Qualite(O_{Rest}) \geq Qualite(O_{Init})$$

2.2.3 Approches de restructuration

Des travaux récents se sont intéressés à la restructuration des ontologies et ont proposé des approches pour le traitement de ce problème dans différentes perspectives. Ces approches, regroupées selon leurs objectifs, sont résumées dans ce qui suit.

Amélioration de la représentation conceptuelle d'une ontologie

Dans la même perspective de notre travail, les approches de ce groupe ont pour but d'améliorer la représentation conceptuelle d'une ontologie.

L'approche proposée par (Gailly et Poels, 2007) utilise des principes de l'IO pour la réingénierie de l'ontologie *Resource Event Agent* (REA) afin qu'elle soit plus appropriée à la modélisation, guidée par les ontologies, des processus d'affaires. Les auteurs se basent sur le processus de restructuration proposé dans la méthode d'IO, Methondology.

Le processus consiste en deux étapes : analyse et synthèse. Lors de la phase d'ana-

lyse, l'hiérarchie de classes, les définitions de concepts et les axiomes sont évalués. Cette phase d'analyse est effectuée en utilisant une des méthodes d'évaluation d'ontologies existantes. Le résultat de l'étape d'analyse est exploité par l'étape de synthèse pour respécifier la conceptualisation. La démarche des auteurs consiste à utiliser une représentation de base de l'ontologie REA (diagramme de classes UML) et appliquer ensuite un certain nombre d'opérations afin d'inclure une axiomatisation plus complète du domaine. Des opérations telles que : ajout de certaines nouvelles classes, relations et cardinalités ; rendre les définitions des relations de concepts plus explicites, complètes et consistantes ; etc. Les classes rajoutées sont des spécialisations de classes existantes ; des spécialisations d'associations sont également rajoutées au modèle.

La phase de restructuration donne lieu à une nouvelle représentation conceptuelle de l'ontologie REA. Cette nouvelle représentation est proposée comme un modèle générique qui peut être instancié pour créer des modèles d'affaires valides. Les auteurs ont démontré les effets des améliorations proposées sur la modélisation d'un processus d'affaires guidée par l'ontologie REA.

L'approche proposée est orientée vers un cas spécifique d'ontologie et elle n'est donc pas généralisable. De plus, les opérations de restructuration sont faites de façon manuelle.

Les auteurs dans (Kehagias *et al.*, 2008; Kehagias *et al.*, 2010) ont présenté une méthodologie formelle dont le but est de fournir un ensemble de critères et de bonnes pratiques pour la restructuration et le raffinement des ontologies. Dans leur travail, les auteurs ont commencé par présenter un ensemble de critères et introduit ensuite une approche méthodologique pour l'application de ces critères à des ontologies existantes. Ils ont démontré l'application de la méthodologie proposée sur une étude de cas spécifique, à savoir la restructuration de six ontologies

qui ont été développées dans le contexte du projet intégré ASK-IT [4].
Le processus de restructuration a été accompli suivant les critères suivants :

1. *Hiérarchie de concepts (taxonomie)* : (a) les concepts de l'ontologie sont réorganisés par l'exploitation des possibilités de regroupements des concepts similaires, (b) les concepts référant au même niveau d'abstraction sont placés au même niveau de l'hiérarchie, (c) la structure des branches qui sont très différentes par rapport aux autres (profondeur, largeur, etc.) est changée afin d'avoir une hiérarchie plus équilibrée ;
2. *Structure de propriétés :* (a) la restructuration est réalisée sur les propriétés par l'exploitation des possibilités de regroupements des propriétés avec de mêmes domaines formant ainsi une structure hiérarchique, (b) de nouvelles propriétés sont rajoutées pour une meilleure description et clarification de concepts ;
3. *Répétition de concepts ontologiques similaires :* le cas où des concepts ontologiques similaires sont répétés fréquemment dans la structure. Ces concepts pourraient éventuellement être combinés en un seul module et réutilisés quand c'est nécessaire ;
4. *Soustraction de modules :* dans le but de réduire la complexité de l'ontologie et la rendre plus compréhensible, les définitions dupliquées sont éliminées, les concepts similaires sont supprimés ou fusionnés et les propriétés qui n'ont jamais été utilisées sont supprimées ;
5. *Documentation/Visualisation :* la documentation de l'ontologie est améliorée par l'ajout de commentaires ;
6. *Définition de domaine/co-domaine des propriétés :* plusieurs propriétés peuvent exister avec des domaines/co-domaines indéfinis, ce qui peut engendrer des

4. www.ask-it.org

inconsistances lors de l'utilisation de l'ontologie. Après la restructuration, le domaine/co-domaine est spécifié pour chaque propriété ;

7. *Restrictions de disjonction :* Il s'agit de reconsidérer attentivement les restrictions de disjonction pour les concepts au sein de l'ontologie (ex. les sous concepts ne sont pas toujours disjoints) ;
8. *Conventions de nommage :* pour des raisons de clarté, les concepts et les propriétés sont renommés suivant un même style de nommage.

Nous trouvons cette approche intéressante pour deux raisons :

1. Les critères proposés couvrent des anomalies qui touchent pratiquement tous les éléments ontologiques et dont leur correction assure le respect des exigences auxquelles doit répondre un modèle ontologique.
2. La restructuration des propriétés sous forme de taxonomie, un aspect qui semble ne pas être traité par les autres approches.

Cependant, on lui reproche le fait que la restructuration consiste en des remaniements locaux faits manuellement ignorant ainsi les effets de ces remaniements sur la structure globale de l'ontologie.

Le travail dans (Wimmer, 2009) consiste à proposer un framework pour la génération d'ontologies à partir de schémas conceptuels antérieurs. Ce framework est instanciable pour divers scénarios de réingénierie de schémas conceptuels et permet de générer des ontologies avec des structures et sémantiques améliorées, comparées aux schémas originaux, par l'exploitation de l'expressivité des langages ontologiques. Ce framework permet également la migration automatique des données à partir des schémas vers des instances des ontologies générées. Pour produire des ontologies riches sémantiquement à partir des schémas existants, les auteurs proposent un processus semi-automatique pour remédier aux limitations des approches de génération automatique existantes.

Le processus inclut deux phases : dans la première « Transformation automatique », une version préliminaire d'une ontologie est générée automatiquement à partir du schéma considéré ; dans la deuxième phase « Validation et restructuration », la version préliminaire de l'ontologie est enrichie sémantiquement et améliorée structurellement.

La génération automatique de l'ontologie lors de la première phase se base sur la définition des correspondances entre le langage du schéma et le langage de l'ontologie. Ces correspondances des éléments entre les deux langages (source et cible) sont implémentées comme des règles de transformation du schéma. Le framework offre également lors de cette phase des heuristiques qui sont appliquées afin de traiter les déficiences existantes dans le schéma et par conséquent améliorer automatiquement l'ontologie initiale générée.

Dans la deuxième phase, l'utilisateur doit valider manuellement le résultat d'application des heuristiques sur l'ontologie générée et ensuite continuer sa restructuration afin de résoudre les anomalies non traitées par les heuristiques. Les opérations de restructuration sont faites manuellement et consistent par exemple à : introduire une nouvelle classe comme abstraction factorisant des propriétés communes ; introduire un nouvel élément ontologique (classe/propriété) ; remplacer un élément ontologique existant comme remplacer un attribut par une relation entre deux classes ; changer le type d'un attribut ; etc. Le framework a été appliqué dans des scénarios d'intégration variés. Les auteurs ont montré l'exécution du framework pour la génération de diagrammes de classes UML à partir de DTDs (*Document Type Definitions*).

La méthode d'IO **NeOn** (Suarez-Figueroa *et al.*, 2012) décrite dans la section 2.1.4 propose un scénario pour la restructuration des ontologies, *scénario 8 : Restructuration des ressources ontologiques.* Ce scénario est utilisé dans les cas où les connaissances contenues dans le modèle conceptuel du réseau d'ontologies

doivent être corrigées et réorganisées pour obtenir un réseau qui couvre les exigences des ontologies.

Les développeurs doivent réaliser l'activité de restructuration pour modifier le réseau d'ontologies en cours de construction après l'activité de conceptualisation. L'activité de restructuration peut être réalisée par l'exécution des sous-activités suivantes qui peuvent être combinées les unes avec les autres de diverses manières :

1. *activité de modularisation d'ontologie :* les développeurs créent différents modules d'ontologies afin de faciliter la réutilisation des connaissances incluses dans le réseau ;
2. *activité d'élagage d'ontologie :* les développeurs élaguent les branches des taxonomies incluses dans le réseau d'ontologies qui sont considérées comme non nécessaires par rapport aux exigences des ontologies ;
3. *activité d'enrichissement d'ontologie :* cette activité peut être réalisée par l'exécution de l'une des deux sous-activités suivantes : *activité d'extension d'ontologie* dans laquelle les développeurs étendent le réseau d'ontologies en incluant (en largeur) de nouveaux concepts et relations ; et *activité de spécialisation d'ontologies* dans laquelle les développeurs rendent plus spécifiques les branches du réseau d'ontologies qui nécessitent plus de granularité en incluant des concepts et des relations plus spécifiques.

Les auteurs proposent d'appliquer cette activité de restructuration des ontologies indépendamment des autres activités/processus ou bien faisant partie du processus de réingénierie des ressources ontologiques. Le résultat de cette activité est un modèle conceptuel du réseau d'ontologies qui représente le domaine concerné.

Le travail dans (Costa *et al.*, 2013) propose une approche de transformation de l'ontologie DAO (*Drosophila Anatomy Ontology*) en une ontologie riche en définitions formelles et textuelles dans laquelle la majorité des classifications sont automatisées. Le travail présente une revue du contenu de la DAO, les patterns

utilisés dans sa formalisation et ses diverses mises en application.
Dans le but d'améliorer la complétude et l'exactitude des classifications dans l'ontologie DAO, les auteurs proposent de la restructurer afin de réduire la classification d'héritage multiple déclaré. Ceci a été réalisé par l'ajout d'une spécification formelle des propriétés de classes et ensuite l'utilisation de cette spécification pour inférer une large proportion de la classification (générer automatiquement la nouvelle classification). En outre, les auteurs ont rajouté des définitions textuelles bien référencées aux classes ainsi que des declarations de disjonctions afin de détecter des erreurs de base.
Les principaux axes de classifications ciblés par la formalisation sont les types de relations qui spécifient : partonomie « Part-of » ou une fonction. Les résultats de ce travail ont montré que l'ontologie a été enrichie en classes et axiomes couvrant tous les aspects de Drosophila anatomy. Grâce à la restructuration, environ la moitié des classifications dans la première version de l'ontologie sont maintenant inférées plutôt que déclarées. La restructuration approfondie dont le but est de détecter des erreurs et d'inférer une classification d'héritage multiple a amélioré considérablement l'exactitude et l'utilité de l'ontologie DAO comme une référence interrogeable par des requêtes anatomiques.
C'est la seule approche, parmi celles étudiées dans cette thèse, qui effectue un remaniement global et génère automatiquement la nouvelle structure de l'ontologie initiale, ce qui rejoint la perspective de notre travail. Cependant, l'approche est proposée pour un cas très spécifique, et elle n'est donc pas généralisable.

Amélioration du résultat d'alignement, d'intégration et de modularisation d'ontologies

Dans un autre objectif, les auteurs dans (Šváb-Zamazal *et al.*, 2008) ont proposé un ensemble d'opérations de restructuration pour corriger les erreurs de conception liées à la structure de deux ontologies avant leur alignement. Le but de leur

travail est de comparer le résultat de l'alignement de deux ontologies avec le résultat de l'alignement de ces mêmes ontologies après leurs restructurations.

La technique proposée permet de détecter certaines erreurs ontologiques en se basant sur la détection des patterns de noms dans la structure de l'ontologie. Une fois ces erreurs détectées, trois opérations génériques de restructuration peuvent être appliquées sur chacune des ontologies : RN (*renaming operation*), RS (*restructuring operation*) et ADD (*operation of adding a concept*). Enfin, le résultat de l'alignement des deux ontologies restructurées sera comparé avec le résultat de l'alignement de ces mêmes ontologies dans leurs formes originales.

L'approche décrite pour la détection des patterns se base sur la notion de « *named structural cluster* » ; trois types de patterns ont été présentés. Le premier SE (*matching siblings with non-matching parent*) : ce pattern représente les cas où deux (ou plusieurs) enfants d'un concept n'ont pas le même nom de tête (*head noun*) que leur parent, mais ils ont le même nom de tête entre eux-mêmes. Ceci peut être dû par exemple à des erreurs de modélisation ou tout simplement à des nommages mal choisis. Pour résoudre ce type de problème, les auteurs proposent l'opération de restructuration RN qui permet de renommer les enfants d'un même concept correctement. Deux options peuvent être appliquées : soit ajouter le nom de tête du parent aux noms de tête de chaque enfant ou bien remplacer carrément le nom de tête de chaque enfant par le nom de tête du concept parent.

Le deuxième pattern hE (*plain head noun*) : ce pattern représente les cas où deux (ou plusieurs) enfants d'un même concept n'ont pas le même nom de tête que leur parent, mais ils ont le même nom de tête entre eux et l'un d'eux possède un nom de tête simple et pas d'autres mots en plus. Dans ce cas là, les auteurs proposent de faire l'opération RS qui permet de déplacer ce concept enfant (ayant un nom de tête simple) dans la hiérarchie et le mettre comme une sous-classe d'un concept ayant le même nom de tête.

Le dernier pattern est ME (*matching outlier*) : ce pattern représente les cas où

un concept partage un même nom de tête avec un autre groupe de concepts dont il n'est pas un descendant. Pour le traitement de ce problème, les auteurs proposent d'appliquer l'opération ADD qui consiste à créer un nouveau concept dans la taxonomie qui va regrouper les concepts ayant le même nom de tête. Pour résumer, toutes les erreurs de conception qui peuvent être détectées au niveau des ontologies par les trois types de patterns présentés peuvent être corrigées par les trois types d'opérations RN, RS et ADD.
Les auteurs ont mené des expérimentations sur sept ontologies de la collection OntoFarm [5] concernant le domaine de l'organisation des conférences. Les patterns sont détectés automatiquement et les ontologies sont restructurées manuellement. Les auteurs ont choisis cinq paires d'ontologies sur lesquelles ils ont appliqué trois outils d'alignement avant et après leur restructuration. Ces expérimentations ont mené à la conclusion que la correction des erreurs existantes au niveau des ontologies à aligner a eu un effet positif et a permis d'améliorer le résultat de leur alignement.

Dans cette même idée, (Behkamal *et al.*, 2010) se base sur le travail de (Šváb-Zamazal *et al.*, 2008) et propose l'utilisation de techniques de détection de patterns et de restructuration pour améliorer la performance d'un outil d'alignement ASMOV (*Automated Semantic Matching of Ontologies with Verification*). Les auteurs proposent d'ajouter une phase de prétraitement au système ASMOV afin d'analyser les ontologies en entrée (à aligner). Cette analyse se base sur la détection de certains patterns lexicaux et structurels pour résoudre les problèmes causés par la diversité des ontologies. Ensuite, des opérations de restructuration sont appliquées sur ces patterns pour établir des ontologies uniformes afin d'améliorer les résultats d'alignement.
La détection des patterns lexicaux se base sur l'analyse des noms d'entités en

5. http ://nb.vse.cz/svatek/ontofarm.html

particulier les classes d'ontologies OWL (*Ontology Web Language*) afin d'établir un même nom à partir des différents styles de nommage utilisés dans chacune des ontologies. Pour résoudre ce problème, les auteurs proposent d'appliquer une des opérations de restructuration définies dans (Šváb-Zamazal *et al.*, 2008), RN (*renaming operation*), pour renommer les classes dans une ontologie en considérant les noms de ces classes dans l'ontologie paire qui ont la même structure taxonomique. Le renommage de classes est avantageux pour le calcul de la similarité lexicale utilisée dans ASMOV.

Un deuxième pattern est basé sur le fait que la structure taxonomique des ontologies est souvent variée et confuse (hiérarchies et granularités différentes pour la définition des entités ontologiques de même domaine). Pour la détection de ces patterns structurels, les auteurs analysent la structure et les relations de subsomption de l'ontologie. Ils utilisent ensuite l'opération de restructuration RS (*restructuring operation*), définie dans (Šváb-Zamazal *et al.*, 2008), pour assimiler les caractéristiques structurelles de cette ontologie par la considération des relations père-fils et des granularités variées qui sont utilisées dans l'ontologie paire. L'idée est de transformer une partie d'une ontologie dans une autre ontologie. Ceci sera avantageux pour l'outil d'alignement ASMOV qui se base sur une méthode de calcul de la similarité structurelle (relationnelle ou hiérarchique) combinant les similarités entre les parents et les enfants des entités en cours de comparaison.

Les expérimentations ont été réalisées sur six paires d'ontologies à partir de « *Conference Track* » qui font partie de OAEI (*Ontology Alignment Evaluation Initiative*) [6].

Le travail dans (Torniai *et al.*, 2013) décrit l'approche et les stratégies appliquées lors du processus d'intégration et de modularisation de deux ontologies en un ISF (Integrated Semantic Framework). Les deux ontologies ont été développées dans

6. http ://oaei.ontologymatching.org/

le cadre des projets eagle-i[7] et VIVO[8] dans le but de représenter et cataloguer les ressources de recherche ainsi que les profils de chercheurs respectifs.
L'approche consiste principalement à faire l'alignement et l'intégration de ces deux ontologies afin de produire un ISF sous forme de modules distincts qui vont remplacer ces dernières dans les applications les utilisant. De plus, ces modules peuvent être utilisés de façon indépendante ou en combinaison avec d'autres applications dans différents domaines de recherche.
La phase de restructuration consiste en la création de nouveaux fichiers OWL qui représentent le nouveau contenu réutilisé et restructuré. Ces fichiers forment conjointement la base du ISF restructuré et modularisé. Cette approche offre la possibilité de faire des choix au niveau d'entités et d'axiomes, de revoir ces choix avec l'équipe des développeurs, et enfin de dériver un nouveau fichier OWL stable durant ce processus. Les opérations de restructuration peuvent consister par exemple à : faire migrer des propriétés d'un module à un autre ; regrouper un ensemble de propriétés retrouvées dans un même contexte sous forme d'une entité ; spécialiser une entité (rajouter des sous classes) ; etc.
La majorité des stratégies et outils ont été développés pour un cas spécifique (besoins particuliers du travail des auteurs). Cependant, ils peuvent être adaptés et réutilisés dans d'autres situations qui nécessitent de comparer et d'intégrer différentes sources d'ontologies.

Analyse syntaxique des ontologies avec règles

Dans l'approche proposée par (Baumeister et Seipel, 2006), les auteurs proposent de faire une analyse syntaxique des ontologies avec règles pour détecter des anomalies. Cette analyse se base sur l'investigation d'un sous ensemble de OWL-DL

7. www.eagle-i.net

8. www.vivoweb.org

prenant en considération les règles, les relations « is-a » et certaines caractéristiques de propriétés (transitivité, complément et disjonction).
Les anomalies discutées sont :

1. *redondance*, due à une duplication (définitions identiques de classes, propriétés ou règles) ou subsomption entre des règles, les auteurs parlent aussi des cas de redondances des atomes au niveau de la partie condition d'une règle ou bien la non satisfiabilité de la partie condition d'une règle ;
2. *circularité* dans des définitions au niveau des règles ou de la taxonomie ;
3. *inconsistance*, à cause des définitions contradictoires ou des erreurs de partitions ; et
4. *déficience*, comme les cas des classes ou propriétés qui ne sont jamais utilisées dans le domaine réel de l'ontologie, les chaines d'héritage, les classes ayant un nombre relativement important de propriétés en commun, etc.

Les auteurs ont donné des définitions formelles de ces anomalies. Ces définitions seront utilisées par des mesures pour détecter les anomalies au niveau de l'ontologie. Avec la description d'une anomalie donnée, les auteurs ont décrit les étapes de restructuration à suivre pour éliminer cette anomalie, par exemple créer une nouvelle classe, créer une nouvelle relation, relier les enfants d'une classe supprimée avec les parents de cette classe, changer les règles correspondantes, etc.
L'approche est bien généralisable mais uniquement pour les ontologies avec règles. Elle a l'avantage de couvrir la majorité des anomalies définies à la section 2.1.6. La détection des anomalies est automatique ; cependant, les opérations de restructuration sont manuelles.

Intégration des données au web sémantique

Dans (Ostrowski, 2008), l'auteur a proposé une approche dynamique pour la restructuration des ontologies utilisée principalement dans l'intégration des données

au web sémantique. La méthodologie se base sur le modèle de l'ontologie (TBOX et ABOX) ainsi que sur l'ensemble des changements qu'on veut effectuer sur cette ontologie. Les changements requis vont guider la définition d'un ensemble de règles de transformation à appliquer pour apporter les modifications nécessaires. Ces règles sont utilisées pour supporter les opérations suivantes : supprimer des propriétés, des relations et/ou des instances qui existent déjà dans l'ontologie ; inférer de nouvelles règles ; inférer de nouvelles données ; etc.). Une fois les classes, propriétés et relations sont mises à jour, l'affectation des instances sera modifiée en conséquence et enfin des requêtes seront appliquées sur les données pour tester la validité du fonctionnement original de l'ontologie. L'implémentation de cette approche utilise l'API Jena [9] avec le langage de programmation interactif JRuby [10] (Java based version of Ruby) pour supporter le développement dynamique.
L'approche ne traite pas les anomalies existantes dans l'ontologie mais plutôt propose une restructuration pour adapter le modèle ontologique aux changements requis.

Comparaison sémantique entre versions d'ontologies

Dans un but de comparaison sémantique et de détection de changements entre des versions d'ontologies OWL-DL, les auteurs dans (Gröner et Staab, 2010) se sont basés sur une classification de patterns de restructuration de modèles, proposée en ingénierie de logiciels, pour tenter d'identifier ces patterns dans des ontologies OWL en utilisant un raisonnement LD (logique de description).
Les auteurs ont commencé par décrire en détails les patterns de restructuration considérés. Ils ont ensuite proposé des algorithmes basés sur l'utilisation du raisonnement logique de description pour la détection de ces patterns.

---

9. http ://jena.sourceforge.net/

10. http ://jruby.org/

Dans leur approche, un pattern de restructuration consiste en ces éléments : (1) le nom du pattern ; (2) la description du problème qui caractérise la structure modélisée d'une ontologie et indique quand ce pattern est appliqué ; (3) la solution qui décrit comment le problème est ou doit être résolu. Elle contient les étapes nécessaires pour réaliser la restructuration ; et (4) un exemple démontrant les détails techniques de cette restructuration incluant les changements appliqués à l'ontologie.

Les patterns sont divisés en trois groupes :

1. *ajout et suppression de classes* qui inclut 3 sous-groupes : le premier sous-groupe extrait une classe à partir d'une classe existante (un ensemble de propriétés d'une classe existante seront déplacées vers une nouvelle classe) ; le deuxième sous-groupe supprime une classe et déplace les propriétés vers une autre classe ; le troisième sous-groupe représente des changements hiérarchiques comme par exemple fusionner des sous et super classes en une seule nouvelle classe ;
2. *déplacement des restrictions de propriétés* qui contient trois sous-groupes : dans le premier sous-groupe, des restrictions de propriétés d'une classe sont déplacées vers une classe référencée ; dans le deuxième sous-groupe, les restrictions de propriétés d'une classe sont déplacées vers une super classe existante ; le troisième sous-groupe couvre le mouvement des restrictions de propriétés d'une classe vers une sous-classe existante ;
3. *modification des restrictions de propriétés* qui inclut les deux sous-groupes suivants : ajout et suppression des références inverses et changement des restrictions de cardinalités de propriétés existantes.

Pour la validation, les auteurs ont choisi deux ontologies DOLCE Lite Plus [11] et

11. http ://loa-cnr.it/DOLCE.html

OBI [12]. Ils ont changé la version originale de chaque ontologie par l'ajout et la suppression de classes, propriétés et axiomes en concordance avec la description des patterns de restructuration définis. Ils ont appliqué ensuite leur approche pour identifier les restructurations faites.

2.2.4 Analyse critique des approches existantes

Après avoir fait une synthèse des principales approches de restructuration des ontologies existantes dans la littérature (voir tableau 2.2), nous discutons dans cette section les limites de ces approches, en particulier les problèmes liés à l'ingénierie ontologique non encore traités par les travaux développés.

12. http ://obi-ontology.org/page/Main_Page

Tableau 2.2 Synthèse des approches de restructuration existantes.

Approche	Objectif	Anomalies	Démarche de restructuration		
			Formalisat.	Opérations	Automat.
(Gailly et Poels, 2007)	Améliorer la représentation conceptuelle de l'ontologie REA	Problèmes issus de l'analyse de l'hiérarchie de classes, des définitions de concepts et des axiomes	Non formelle	Ajouter classes, relations et cardinalités, spécialiser classes et associations existantes, etc.	Manuelle
(Kehagias et al., 2008; Kehagias et al., 2010)	Fournir des critères pour la restructuration et le raffinement des ontologies	Erreurs dans l'hiérarchie de concepts, propriétés malstructurées, concepts similaires répétés, etc.	Formelle	Faire des regroupements de concepts et de propriétés, ajouter nouvelles propriétés, rendre branches de l'hiérarchie plus équilibrées, etc.	Manuelle
(Wimmer, 2009)	Proposer un framework pour la génération d'ontologies à partir de schémas conceptuels antérieurs	Déficiences existantes dans le schéma de départ	Non formelle	Introduire nouvelle abstraction factorisant des propriétés communes, remplacer un attribut par une relation entre deux classes, etc.	Semi-automatique
(Suarez-Figueroa et al., 2012)	proposer un scénario pour la restructuration des ontologies	Connaissances à corriger et réorganiser par rapport aux exigences du réseau d'ontologies	Non formelle	Créer modules d'ontologies, élaguer branches des taxonomies non nécessaires, étendre ontologies avec de nouveaux concepts et relations, etc.	Manuelle
(Costa et al., 2013)	Transformer l'ontologie DAO en une ontologie riche en définitions formelles et textuelles	Erreurs d'inconsistances, incomplétude et erreurs de classifications	Formelle	Spécification formelle des classes et propriétés pour inférer la nouvelle classification	Automatique

Approche	Objectif	Anomalies	Démarche de restructuration		
			Formalisat.	Opérations	Automat.
(Šváb-Zamazal et al., 2008)	Améliorer le résultat d'alignement de deux ontologies	Erreurs de conception détectées par 3 patterns de noms (SE, hE, ME)	Définitions formelles des patterns	Opérations (RN, RS, ADD) pour résoudre chaque type de problème détecté par chaque pattern	Manuelle
(Behkamal et al., 2010)	Améliorer la performance d'un outil d'alignement ASMOV	Problèmes détectés par des patterns lexicaux et structurels	Non formelle	Appliquer l'opération de renommage (RN) ou l'opération de restructuration (RS)	Manuelle
(Torniai et al., 2013)	Intégration et modularisation de deux ontologies	Problèmes résultants de fusion et modularisation de deux ontologies	Non formelle	Migrer propriétés entre modules, regrouper propriétés sous forme d'entité, etc.	Manuelle
(Baumeister et Seipel, 2006)	Analyse syntaxique des ontologies avec règles	Redondance, circularité, inconsistance et déficience	Définitions formelles des anomalies	Créer nouvelle classe/relation, supprimer classe redondante, etc.)	Manuelle
(Ostrowski, 2008)	Intégration des données au web sémantique	Changements requis	Définition d'un ensemble de règles	Appliquer les règles pour supprimer propriété, relation, inférer nouvelles classes, propriétés, etc.)	Automatique
(Gröner et Staab, 2010)	Comparaison sémantique entre des versions d'ontologies	Problèmes détectés par un ensemble de patterns	Formelle	Ajouter et supprimer classes, déplacer restrictions de propriétés, modifier restrictions de propriétés, etc.	Manuelle

Premier constat

Les approches existantes ont abordé la restructuration des ontologies dans des situations spécifiques et avec des objectifs fort divergents. Cependant, il n'existe toujours pas de méthodologie bien établie qui couvre tout le processus de restructuration et qui permet de faire un remaniement global de la structure de l'ontologie. De plus, les protocoles de validation sont souvent orientés vers une application particulière et ne sont donc pas généralisables. Cela rend difficile leur comparaison.

Le tableau 2.3 fait une comparaison de ces approches par rapport à un certain nombre de critères que nous avons choisis. Les approches sont évaluées, d'une part, sur leur traitement ou non d'une anomalie donnée parmi celles définies à la section 2.1.6 : détectée et corrigée (✓), non traitée (-) ou non précisé (?), la colonne « autres » spécifie si d'autres types de problèmes sont traités (imperfections selon le contexte, changements requis, erreurs détectées par des patterns, etc.) ; et, d'autre part, sur la démarche de restructuration. L'idée est de préciser pour chaque approche, si (1) la restructuration consiste à faire des remaniements locaux (RL) ou un remaniement global de l'ontologie (RG) ; (2) les opérations de restructuration sont formelles (F) ou non formelles (NF) ; (3) ces opérations sont automatiques (A), semi-automatiques (SA) ou manuelles (M) ; (4) l'approche est généralisable (G), spécifique -ontologie de base- (S) ou spécifique mais applicable sur d'autres ontologies (S–»G).

Deuxième constat

Pas de bonne compréhension de ce que le processus de restructuration doit couvrir en tant que tâches de détection de problèmes et tâches de correction, et donc il manque un cadre formel pour la résolution conjointe de ces deux types de tâches. En outre, les tâches de restructuration proposées par les approches étudiées impliquent des remaniements locaux ; ce qui constitue un défi pour les concepteurs d'ontologies du fait que ces remaniments, comme la création de nouveaux éléments et le déplacement de ceux existants, pourraient avoir des répercussions sur la structure globale de l'ontologie.

Troisième constat

Les travaux présentés s'attachent plus à un seul niveau de la restructuration, à savoir la correction des erreurs de conception liées à la structure de l'ontologie. De plus, les opérations proposées pour la correction de ces erreurs sont le plus souvent manuelles. Le deuxième niveau, celui concernant la découverte des connaissances

manquantes, n'est pas bien pris en compte par les approches existantes.

Approche	Anomalies traitées					Démarche de restructuration			
	Incons.	Incomp.	Red.	Défic.	Autres	Type	Formalisat.	Automatisat.	Généralisat.
(Gailly et Poels, 2007)	✓	✓	?	-	✓	RL	NF	M	S
(Kehagias *et al.*, 2008; Kehagias *et al.*, 2010)	✓	-	✓	✓	✓	RL	F	M	S–»G
(Wimmer, 2009)	?	✓	?	✓	✓	RL	NF	SA	G
(Suarez-Figueroa *et al.*, 2012)	?	✓	?	-	✓	RL	NF	M	S–»G
(Costa *et al.*, 2013)	✓	✓	?	-	✓	RG	F	A	S
(Šváb-Zamazal *et al.*, 2008)	✓	-	-	-	✓	RL	NF	M	G
(Behkamal *et al.*, 2010)	✓	-	-	-	✓	RL	NF	M	G
(Torniai *et al.*, 2013)	✓	-	✓	✓	✓	RL	NF	M	S–»G
(Baumeister et Seipel, 2006)	✓	-	✓	✓	-	RL	NF	M	G
(Ostrowski, 2008)	-	-	-	-	✓	RL	NF	A	G
(Gröner et Staab, 2010)	✓	-	?	✓	✓	RL	F	M	G

Tableau 2.3 Comparaison des approches de restructuration étudiées.

Pour toutes ces raisons, nous pensons que l'élaboration d'une méthodologie pour la restructuration des ontologies s'avère nécessaire. Une méthodologie qui permettra de :

1. couvrir tout le processus de restructuration ;
2. faire une réorganisation complète de l'ontologie en prenant en considération les deux niveaux de restructuration : (1) la détection et la correction des erreurs de conception déjà existantes dans l'ontologie ; et (2) l'identification des connaissances qui manquent et leur intégration dans l'ontologie initiale ;
3. avoir un degré d'automatisation le plus élevé.

2.3 Conclusion

Ce chapitre nous a permis de constater qu'un modèle ontologique est sujet à des erreurs ou des anomalies de conception résultant d'un certain nombre d'opérations qui peuvent mener à la détérioration de sa qualité. Une étude critique a montré les limites des méthodes de restructuration dans le domaine, et l'intérêt d'une orientation de la recherche vers la formalisation et l'automatisation du processus de restructuration pour la construction d'ontologies mieux structurées et plus complètes. Le chapitre suivant présente notre approche pour la restructuration des ontologies basée sur un cadre formel qui est « l'Analyse relationnelle de concepts ».

CHAPITRE III

CADRE DE RESTRUCTURATION DES ONTOLOGIES PAR L'ARC

Ce chapitre présente notre cadre pour la restructuration des ontologies par l'ARC (Analyse relationnele de concepts), une extension de l'AFC (Analyse formelle de concepts). L'AFC nous fournit un cadre structurel et formel qui va supporter le processus global de la restructuration. Dans cette approche, l'AFC est utilisée comme une technique de catégorisation conceptuelle (conceptual clustering).

L'approche décrite dans ce chapitre, à laquelle nous avons contribué, a été développée par notre équipe impliquant plusieurs chercheurs dans le cadre du projet *PRIOWS* (Projet de recherche en ingénierie ontologique et web sémantique).

Nous commençons par examiner le choix de l'AFC/ARC comme cadre théorique et présenter les notions fondamentales nécessaires à sa compréhension. Ensuite, nous donnons la description détaillée des différentes phases constituant le processus général de la restructuration. Finalement, nous expliquons comment notre cadre traite certaines des anomalies identifiées à la section 2.1.6.

3.1 Pourquoi l'AFC/ARC?

Le choix de l'AFC/ARC comme cadre théorique pour l'approche de restructuration est motivé par le besoin de mettre à plat l'hiérarchie de l'ontologie initiale et tenter ensuite de reconstruire une nouvelle hiérarchie qui factorise mieux les descriptions et améliore le niveau d'abstraction. Ce choix s'appuie également sur le fait que c'est un cadre formel avec des méthodes universelles (Ganter et Wille, 1999) qui a été largement utilisé dans de nombreux domaines, et les résultats témoignent de son apport dans la construction des hiérarchies de classes (Dao *et al.*, 2004) et de concepts (Stumme et Maedche, 2001; Cimiano *et al.*, 2005; Nanda *et al.*, 2006; Bendaoud *et al.*, 2008; Rouane-Hacene *et al.*, 2008) ainsi que dans la restructuration des modèles (Godin et Mili, 1993; Godin *et al.*, 1998; Snelting, 2000; Godin et Valtchev, 2005; Rouane-Hacene *et al.*, 2007).

En réingénierie des modèles, l'AFC est considérée comme un cadre théorique approprié pour la restructuration des hiérarchies de classes. Il s'agit de faire une redistribution des propriétés d'une hiérarchie existante, typiquement orientée objet, sur un ensemble potentiellement différent de classes organisées sous forme d'un treillis de concepts. Le treillis généré va représenter la nouvelle hiérarchie reconstruite avec une factorisation maximale des propriétés et l'intégration de nouvelles abstractions, potentiellement utiles, oubliées par les concepteurs (Rouane-Hacene *et al.*, 2007).

L'ARC est une extension de l'AFC pour le traitement des données relationnelles. Elle offre une meilleure factorisation au sein d'un modèle, car tous les éléments du modèle sont pris en compte dans le processus de généralisation notamment les associations, et par conséquent des abstractions utiles peuvent être détectées en regroupant les individus non seulement à partir de leurs descripteurs propres mais aussi à partir des liens qu'ils partagent (Rouane-Hacene *et al.*, 2007).

En ingénierie ontologique, les méthodes de l'AFC/ARC ont étés utilisées comme des techniques de catégorisation conceptuelle dans le processus de construction des ontologies (Stumme et Maedche, 2001; Cimiano *et al.*, 2005; Nanda *et al.*, 2006; Bendaoud *et al.*, 2008; Rouane-Hacene *et al.*, 2008). Ces approches de construction basées sur l'AFC ont été proposées pour remédier aux limites des approches « informelles » qui présentent l'avantage d'extraire les termes les plus représentatifs du domaine étudié à partir du corpus utilisé, mais elles peuvent négliger certaines entités qui peuvent s'avérer pertinentes pour le domaine concerné et qui ne sont pas extraites, nécessairement, à partir des textes.

Donc, toute cette performance de l'AFC et de l'ARC dans l'ingénierie ontologique ainsi que dans la restructuration des hiérarchies de classes nous laisse croire qu'elles pourraient donner les résultats escomptés de la restructuration d'une ontologie.

3.2 ARC : Notre cadre théorique

Dans cette section et en premier lieu, nous présentons les définitions mathématiques des notions fondamentales de l'AFC. Nous abordons uniquement quelques notions de base nécessaires pour la compréhension de notre cadre de recherche. Tous les fondements théoriques de l'AFC peuvent être trouvés dans (Ganter et Wille, 1999; Carpineto et Romano, 2004). Deuxièment, nous présentons notre cadre théorique, l'ARC.

3.2.1 Notions fondamentales de l'AFC

L'AFC est définie comme un paradigme d'analyse (Ganter et Wille, 1999) qui s'appuie sur la théorie des treillis (Birkhoff, 1967; Davey et Priestley, 1990). Plus spécifiquement, l'AFC constitue une approche algébrique pour analyser des tableaux

Tableau 3.1 Différences clés avec les notations standards de l'AFC.

Not.	Description	Not.	Description
O	L'ensemble des objets formels (G)	$\mathcal{C}_K^o$	La famille des extensions de K
A	L'ensemble des attributs formels (M)	$\mathcal{C}_K^a$	La famille des intensions de K
$\mathcal{C}_K$	L'ensemble des concepts formels de K ($\mathcal{C}(K)$)	$\mathcal{L}_K$	Le treillis de concepts de $K(\mathcal{B}(\mathcal{K}))$

de données binaires (*contextes formels*). Les contextes représentent un ensemble d'individus (*objets*) décrits par un ensemble de propriétés (*attributs*) via une relation binaire précisant pour un objet donné les propriétés qu'il possède. Cette analyse permet de faire des regroupements d'objets ayant des attributs en commun (*concepts formels*) et de les organiser sous formes de structures conceptuelles (*treillis de concepts formels*).

Les notations utilisées dans cette thèse différent légèrement des notations standards de l'AFC. Les principales différences sont résumées dans le tableau 3.1.

Contexte formel

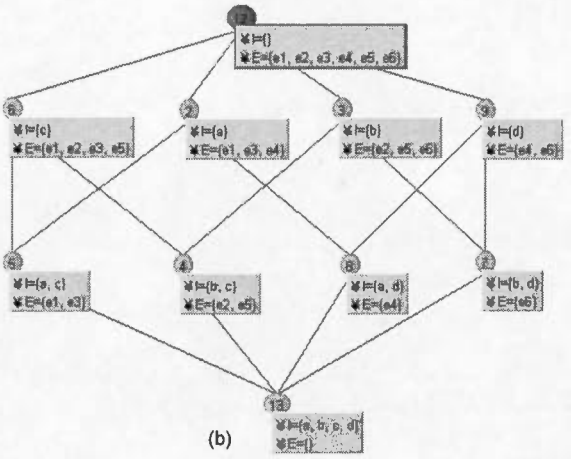
Mathématiquement parlant, un contexte formel est défini comme suit :

Définition 1. Un *contexte formel (mono-valué)* est un triplet $K = (O, A, I)$ où O est l'ensemble des objets formels, A est l'ensemble des attributs formels et $I \subseteq O \times A$ est une relation binaire, appelée *d'incidence*, entre les objets et les attributs, telle que pour $o \in O$ et $a \in A$, $(o, a) \in I$ (notée oIa) signifie que l'objet o possède l'attribut a.

Exemple : La table (a) illustrée à la figure 3.1 présente un contexte mono-valué, appelé « *Entreprises* », où les objets formels $\{e1, e2, e3, e4, e5, e6\}$ représentent

	a	b	c	d
e1	x		x	
e2		x	x	
e3	x		x	
e4	x			x
e5		x	x	
e6		x		x

(a)



(b)

Figure 3.1 *(a)* : Un contexte binaire « Entreprises ». *(b)* : Le treillis de concepts associé.

des *entreprises* et les attributs formels $\{a, b, c, d\}$ représentent certaines caractéristiques commerciales de ces entreprises.

Les contextes formels pluri-valués ont été introduits en AFC (Ganter et Wille, 1999) pour coder les attributs non binaires (ex. numériques, ordinaux, catégoriques, etc.). Un *contexte pluri-valué* est un quadruplet $K = (O, A, V, I)$ où O est l'ensemble des objets, A est l'ensemble des attributs, V est un ensemble de valeurs, et $I \subseteq O \times A \times V$ est une relation ternaire ayant des tuples sous la forme (o, a, v) qui signifie l'objet o a la valeur v pour l'attribut a.

Le traitement d'un contexte pluri-valué passe tout d'abord par une étape de transformation de ce contexte en un contexte binaire (mono-valué) équivalent. Cette étape est assurée par un mécanisme appelé *graduation conceptuelle* (Ganter et Wille, 1999) qui permet de construire un contexte échelle par attribut pluri-valué et ensuite assembler les échelles en un même contexte. Par exemple, l'attribut pluri-valué *nbprod* (nombre de produits) dont les valeurs sont $\{8, 25, 10, 11\}$ peut être transformé en attributs binaires de la forme $nbprod \leqslant 10$ et $nbprod > 10$

Clients	c1	c2	c3	c4
nbprod	8	25	10	11

	nbprod ≤ 10	nbprod > 10
c1	x	
c2		x
c3	x	
c4		x

Figure 3.2 Transformation d'un attribut pluri-valué « *nbprod* » en attributs binaires.

(voir figure 3.2).

Correspondance de Galois dans un contexte formel

Étant donné un contexte formel $K = (O, A, I)$, deux fonctions f et g sont définies sur ce contexte :

Définition 2. La fonction f associe à un ensemble d'objets X, l'ensemble d'attributs Y partagés par tous les objets de X tandis que g est la fonction duale pour les ensembles d'attributs :

$f : \wp(O) \longrightarrow \wp(A)$ tel que : pour $X \in \wp(O), f(X) = X' = \{a \in A \mid oIa \ \forall o \in X\}$
$g : \wp(A) \longrightarrow \wp(O)$ tel que : pour $Y \in \wp(A), g(Y) = Y' = \{o \in O \mid oIa \ \forall a \in Y\}$

Propriété 1. La paire de fonctions (f, g), les deux notées $'$ et résumant les liaisons entre les objets et les attributs dans le contexte, définit *une correspondance de Galois* (Barbut et Monjardet, 1970) entre les deux ensembles ordonnés $\wp(O)$ et $\wp(A)$ du contexte formel $K = (O, A, I)$.

Propriété 2. Les compositions des fonctions f et g : $fog : \wp(A) \longrightarrow \wp(A)$ et $gof : \wp(O) \longrightarrow \wp(O)$ sont les opérateurs de fermeture sur $\wp(A)$ et $\wp(O)$, respectivement. Les deux opérateurs de composition sont notés par $''$.

Un opérateur de fermeture est défini comme suit :

Définition 3. Soit $(E, \leq)$ un ensemble ordonné. On appelle opérateur de fermeture sur l'ensemble E toute application $\varphi : E \rightarrow E$ qui vérifie les trois propriétés suivantes :

- idempotence : $\forall x \in E, \varphi(\varphi(x)) = \varphi(x)$,
- monotonie : $\forall x, y \in E, x \leq y \Rightarrow \varphi(x) \leq \varphi(y)$,
- extensivité : $\forall x \in E, x \leq \varphi(x)$.

Étant donné le contexte formel $K = (O, A, I)$ et l'opérateur de fermeture $''$, un sous ensemble d'objets $X \in \wp(O)$ (respectivement un sous ensemble d'attributs $Y \in \wp(A)$) est fermé si $X'' = X$ (respectivement $Y'' = Y$).

Concept formel

Le couple (X, Y) d'ensembles fermés où X représente un sous-ensemble d'objets et Y un sous-ensemble d'attributs est appelé *concept formel.*

Définition 4. Un *concept formel* d'un contexte K est une paire (X, Y) où $X \subseteq O, Y \subseteq A, X = Y'$ et $Y = X'$. L'ensemble X est appelé *extension* et Y *l'intension* du concept formel.

Exemple : Dans la le contexte de la figure 3.1, le couple $(\{e1, e3\}, \{a, c\})$ est un concept formel, où $\{e1, e3\}$ représente son *extension* et $\{a, c\}$ représente son *intension*, alors que $(\{e1, e3\}, \{a\})$ n'en est pas un.

L'ensemble de tous les concepts formels d'un contexte $K = (O, A, I)$ est noté par C_K. La taille de cet ensemble peut être exponentielle en la taille du contexte. Plusieurs algorithmes ont été proposés pour le calcul de cet ensemble ou encore la relation d'ordre entre les éléments de cet ensemble.

Treillis de concepts formels

La famille C_K des concepts formels d'un contexte $K = (O, A, I)$ est partiellement ordonnée par la relation d'inclusion $\subseteq$ entre intensions/extensions.

La famille des fermés des objets $C^o \subseteq \wp(O)$ et celle d'attributs $C^a \subseteq \wp(A)$ munies de la relation d'ordre partiel $\subseteq$ (inclusion ensembliste) constituent deux treillis complets, notés L^o et L^a. De plus, la fonction $'$ (f ou g) est une bijection entre les deux familles C^o et C^a et définit un isomorphisme entre les treillis respectifs L^o et L^a. À chaque fermé X de C^o correspond un fermé unique Y de C^a, tel que $X' = Y$.

Définition 5. Étant donné (X_1, Y_1) et (X_2, Y_2) deux concepts formels d'un contexte $K = (O, A, I)$. Le concept (X_1, Y_1) est un sous-concept de (X_2, Y_2) (d'une manière équivalente, (X_2, Y_2) est un super-concept de (X_1, Y_1)) si et seulement si $X_1 \subseteq X_2$ ($Y_2 \subseteq Y_1$). On utilise le signe $\leq$ pour exprimer cette relation. On obtient : $(X_1, Y_1) \leq_K (X_2, Y_2) \Longleftrightarrow X_1 \subseteq X_2$ (ou bien $Y_2 \subseteq Y_1$).

Propriété 3. L'ensemble de tous les concepts formels de $K = (O, A, I)$ ordonné par la relation de super-concept (sous-concept) est un treillis complet. Il est noté $L_K = \langle C_K, \leq_K \rangle$ et est appelé le *treillis de concepts* du contexte K.

Le treillis de concepts est muni des opérations dites *infimum* et *supremum* qui sont notées, respectivement, par $\wedge$ et $\vee$.

Théorème 1. L'ordre partiel $L_K = \langle C_K, \leq_K \rangle$ forme un treillis complet où l'infimum et le supremum sont définis comme suit :

- $\wedge_{i=1}^{k}(X_i, Y_i) = (\cap_{i=1}^{k} X_i, (\cup_{i=1}^{k} Y_i)'')$
- $\vee_{i=1}^{k}(X_i, Y_i) = ((\cup_{i=1}^{k} X_i)'', \cap_{i=1}^{k} Y_i)$

Définition 6. La couverture supérieure (respectivement inférieure) d'un concept est l'ensemble de tous ses super-concepts (respectivement sous-concepts).

Exemple : La partie (b) de la figure 3.1 illustre le treillis construit, montré comme un « diagramme de Hasse[1] », à partir du contexte (a) représenté à gauche de

1. Tout ensemble ordonné $(E, \leq)$ peut être représenté de façon schématique par un dia-

la figure. Notons que les *extensions* et les *intensions*, identifiables par les deux ensembles respectifs « E » et « I », sont montrées sur les noeuds représentant les concepts.

Algorithmes de construction des treillis

La construction des treillis est composée principalement des deux tâches suivantes :

- La découverte des concepts à partir d'un contexte formel,
- Le calcul de la relation de couverture représentant l'ordre entre l'ensemble de concepts.

La littérature sur l'AFC offre une panoplie d'algorithmes qui se sont intéressés au traitement de ces deux tâches (Ganter et Wille, 1999). Ces algorithmes diffèrent selon la manière d'acquérir et/ou de traiter les données en entrés et sont généralement regroupés en trois grandes classifications. Les premiers algorithmes de construction des treillis sont *les algorithmes batch* (Ganter, 1984; Bordat, 1986; Nourine et Raynaud, 1999) qui considèrent que les données (contexte formel) sont connues à l'avance et l'évolution de ces données (ajout objet/attribut au contexte) entraîne la reconstruction du treillis à nouveau. Ensuite, *les algorithmes incrémentaux* (Godin *et al.*, 1995; Valtchev, 1999) sont apparus pour remédier au problème de la reconstruction du treillis dans le cadre de contextes dynamiques. Suite à une modification du contexte, ces algorithmes effectuent des mises à jour locales du treillis correspondant. Enfin, *les algorithmes d'assemblage* (Valtchev *et al.*, 2002) qui se basent sur l'apposition/sous-position de contextes et les demi-produits de treillis pour définir une opération binaire sur les treillis complets, appelée *assemblage*, permettant de construire un treillis à partir des deux treillis associés aux deux parties du contexte.

---

gramme appelé **diagramme de Hasse** qui est un graphe où les noeuds sont les éléments de E et les acrs représentent la relation de couverture entre éléments.

	e	f	g	h
p1	x			
p2		x		
p3			x	
p4				x

(a)

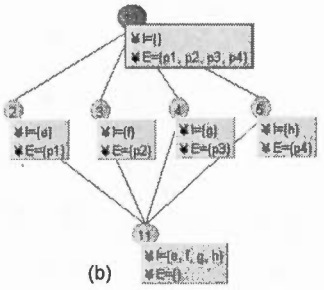


Figure 3.3 *(a) :* Un deuxième contexte binaire « Produits ». *(b) :* Le treillis de concepts associé.

3.2.2 Analyse relationnelle de concepts

Le but de l'ARC (Rouane-Hacene *et al.*, 2013) est d'inférer des relations entre des concepts formels en se basant sur des liens entre des objets formels. En ARC, le codage des données est fait à travers une structure appelée *« Famille de contextes relationnels »*.

Définition 7. Une *Famille de contextes relationnels (FCR)* est une paire (K, R), telle que : $K = \{K_i\}_{i=1,...,n}$ un ensemble de contextes pluri-valués $K_i = (O_i, A_i, I_i)$, un par catégorie, et $R = \{r_k\}_{k=1,...,m}$ un ensemble de relations binaires $r_k \subseteq O_i$ x O_j tels que : $i, j \in \{1, ..., n\}$, avec O_i (domaine de r_k) et O_j (co-domaine de r_k) sont des ensembles d'individus des contextes K_i et K_j, respectivement.

La figure 3.3 montre un deuxième contexte représentant des « Produits » (a) et son treillis associé (b). Un type de lien existant entre les deux contextes *Entreprises* et *Produits* est représenté par une relation inter-contextes *« fab »* comme le montre la partie (a) de la figure 3.4. La relation *« fab »* modélise le lien entre une entreprise et un produit qui exprime la sémantique « une entreprise e fabrique un produit p ».

fab	p1	p2	p3	p4
e1			x	
e2	x			
e3			x	
e4	x	x		
e5			x	x
e6	x	x	x	

(a)

	a	b	c	d	fab:c2	fab:c3	fab:c4	fab:c5	fab:c10
e1	x		x				x		x
e2	x		x		x				x
e3		x		x			x		x
e4	x		x	x	x	x			x
e5	x		x				x	x	x
e6					x	x	x		x

(b)

Figure 3.4 *(a) :* La relation inter-contextes *« fab »*. *(b) :* Extension du contexte des *« Entreprises »* en attributs relationnels obtenue par la graduation de la relation *« fab »*.

Traitement des relations en ARC

Pour le traitement d'une FCR, un mécanisme appelé *graduation relationnelle* (Rouane-Hacene *et al.*, 2013) est utilisé pour traduire les liens inter-objets en un ensemble d'attributs relationnels. Pour ce faire, les relations d'une FCR (ex. *« fab »*) sont considérées comme des attributs pluri-valués $r : O_i \longrightarrow \wp(O_j)$ dont les valeurs sont des ensembles d'individus avec des échelles « relationnelles » structurant le domaine de valeurs (les sous-ensembles $r(o)$, $o \in O_i$) d'une relation. Cette structure est traduite en un ensemble d'attributs relationnels qui sont des prédicats décrivant les ensembles d'objets référencés. Ainsi, pour une relation donnée $r : O_i \longrightarrow \wp(O_j)$, de nouveau attributs, de la forme *« r :c »*, sont rajoutés au contexte K_i dont c est un concept de K_j. Un objet $o \in O_i$ aura un attribut *« r :c »* dépendamment de la relation entre son ensemble de liens $r(o)$ et l'ensemble *extension* de c. De plus, ces attributs impliquent des opérateurs de quantification (Rouane-Hacene *et al.*, 2013) : *existentiel*, *universel*, *existentiel avec restriction de cardinalités*, etc. Dans le cas de notre travail, ces attributs relationnels réfèrent systématiquement à un *quantificateur existentiel* ($\exists$) que nous ommettons pour des raisons de lisibilité.

La partie (b) de la figure 3.4 montre le résultat de la graduation de la relation

« fab » sur le contexte des *« Entreprises »* en utilisant le treillis des *« Produits »*. Par exemple : l'objet formel *« e1 »* est décrit par deux nouveaux attributs relationnels *« fab :c4 »* et *« fab :c10 »* puisque le produit *p3* fabriqué par *e1* apparait dans l'extension des concepts $c_{\#4}$ et $c_{\#10}$ dans le treillis des *« Produits »* (voir (b) de la figure 3.3). Cela est interprété comme suit : *l'entreprise* e1 *est liée par la relation* « fab » *à au moins un produit* p *des concepts* $c_{\#4}$ *et* $c_{\#10}$.

Famille de treillis relationnels (FTR)

Le processus global de l'ARC suit une méthode, appelée *« Multi-FCA »* (Rouane-Hacene *et al.*, 2013) qui permet de construire à partir d'une *FCR* un ensemble de treillis, un par contexte, appelé *« Famille de treillis relationnels (FTR) »*. La *FTR* est définie comme un ensemble de treillis dont les concepts reflètent conjointement tous les attributs ainsi que tous les liens partagés entre objets de la FCR.

Étant donné une *FCR = (K, R)*, avec K_i^p qui dénote le contenu de $K_i = (O_i, A_i, I_i)$ à l'étape p. La méthode *« Multi-FCA »* suit un processus itératif comme le montre les étapes suivantes :

- **Étape [0]** : $\mathcal{L}_i^0$ est construit à partir de K_i^0, le contexte obtenu en prenant en compte tous les attributs du contexte initial K_i à l'exception des relations *(r)*. Des graduations sont effectuées au préalable (si nécessaires).
- **Étape [p+1]** : $\mathcal{L}_j^p$ est utilisé comme échelle pour chaque relation $r \subseteq O_i \times O_j$ afin de produire le contexte K_i^{p+1} et ensuite le treillis correspondant $\mathcal{L}_i^{p+1}$.
- **Critère d'arrêt** : La procédure s'arrête quand $\mathcal{L}_i^{n+1}$ est isomorphe à $\mathcal{L}_i^n$ pour tout i, c'est-à-dire le point fixe de la fonction définie par la procédure *Multi-FCA* est atteint. La limite de la suite $\mathcal{L}_i^p$, pour un $i \in [1, .., n]$, est dénotée par $\mathcal{L}_i^\infty$.

Par exemple, la figure 3.5 montre le point fixe des treillis des deux contextes *« Entreprise »* (a) et *« Produits »* (b).

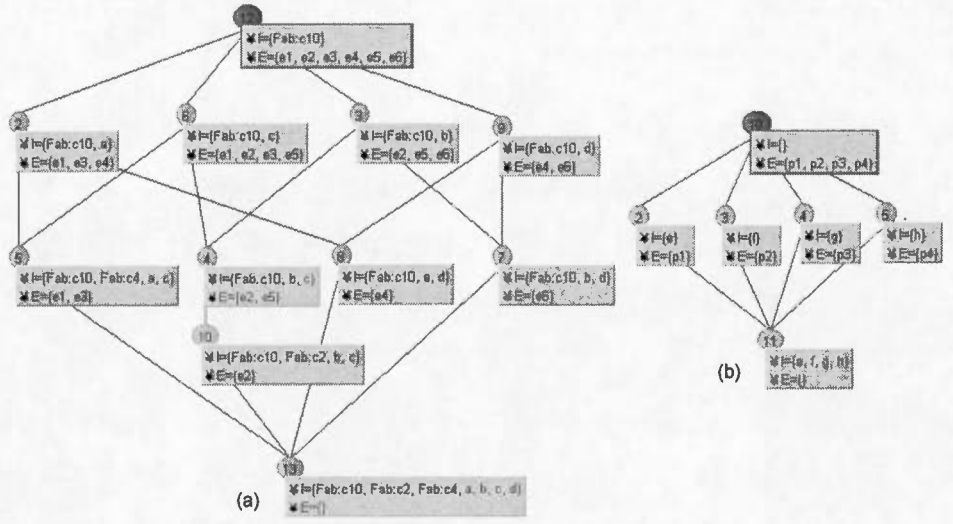


Figure 3.5 Le point fixe des treillis finaux des deux contextes *« Entreprises »* (a) et *« Produits »* (b).

L'AFC possède l'avantage d'être exacte mais présente l'inconvénient de générer des treillis de très grandes tailles qui peuvent contenir un nombre exponentiel de concepts formels, menant à l'existence de beaucoup de concepts superflus (Carpineto et Romano, 2004). L'ARC génère un nombre encore plus grand de concepts formels du fait qu'elle traite des contextes plus riches (FCR).

3.3 Apperçu général de l'approche de restructuration

Étant donnée une ontologie OWL à restructurer, l'approche de restructuration par l'ARC se résume en : (1) coder l'ontologie dans la structure d'entrée de l'ARC; (2) appliquer l'ARC pour construire les treillis correspondants; et (3) traduire l'ensemble des treillis construits en une ontologie restructurée. Avec une telle démarche, deux défis de taille peuvent être identifiés d'orcs et déjà : (1) codage de l'ontologie qui implique tout d'abord la confrontation entre les éléments on-

tologiques, et ensuite leur codage dans le format de données de l'ARC ; et (2) génération de l'ontologie restructurée.

Notons que la génération de l'ontologie restructurée à partir des treillis n'est pas directe. À ce niveau là, nous allons nous retrouver face à un phénomène qu'on reproche souvent à l'AFC, à savoir la génération des treillis de très grandes tailles avec beaucoup de concepts fortuits. Ceci peut nuire à l'interprétabilité des treillis construits dans notre contexte de restructuration.

Notre principale contribution consiste à gérer cette complexité des treillis de concepts issus de l'analyse par le filtrage des éléments pertinents. Le défi ici est de trouver comment, à partir d'une structure complètement formelle et sans accès à la sémantique des entités manipulées [2], sélectionner les concepts formels pertinents qui vont constituer les éléments ontologiques. À ce stade là, nous ne cherchons pas à faire un filtrage des treillis dans l'absolu, mais nous visons plutôt un filtrage informé par l'ontologie de départ.

Notre démarche pour ce filtrage consiste, dans un premier temps, à chercher un moyen pour juger « la pertinence » des concepts formels. Il s'agit de mesurer une pertinence relative pour chaque concept formel au niveau du treillis par rapport au domaine de l'ontologie initiale. Dans un deuxième temps, proposer une technique pour le filtrage en se basant sur les pertinences relatives des concepts formels.

---

2. Notons que l'AFC s'appuie uniquement sur un raisonnement structurel pour faire émerger la sémantique.

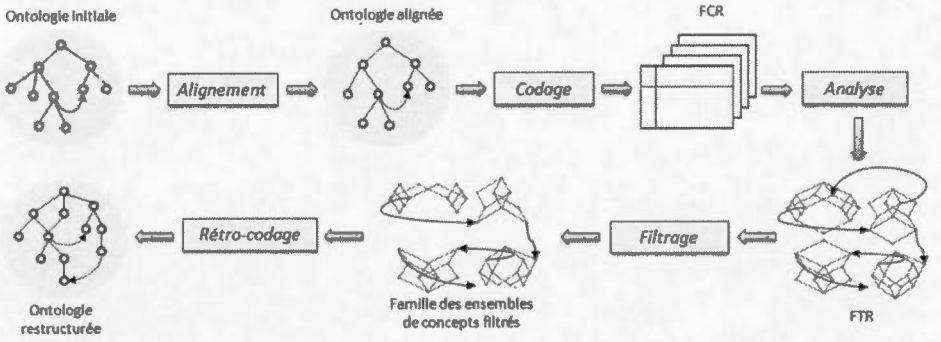


Figure 3.6 Processus général de l'approche de restructuration par l'ARC.

3.4 Processus général de la restructuration des ontologies par l'ARC

Le processus général adopté pour l'approche de restructuration des ontologies par l'ARC (Rouane-Hacene *et al.*, 2010) suit cinq phases principales (voir figure 3.6) : *l'alignement*, *le codage*, *l'analyse*, *le filtrage* et *le rétro-codage*. L'étape *alignement* consiste à faire une confrontation entre les éléments de l'ontologie à restructurer pour l'identification des similarités. Ceci permettra d'éliminer les redondances et d'éviter la duplication du codage des éléments ontologiques similaires. Dans l'étape *codage*, l'ontologie de départ sera transformée en une seule *Famille de contextes relationnels (FCR)* dans laquelle chaque contexte représente un type d'éléments du méta-modèle ontologique (ex. *concepts et rôles*). Les liens existants dans le méta-modèle entre ces deux types d'éléments seront représentés par des relations inter-contextes (ex. *domain, range et owns*). L'étape *analyse* consiste à construire des treillis initiaux (treillis de concepts et de rôles) avec l'AFC. Ensuite, traduire les relations inter-contextes en attributs relationnels suivant le mécanisme de *graduation relationnelle*. Enfin, obtenir des treillis finaux de concepts formels selon le processus itératif ARC. Ce sont les treillis finaux qui fournissent le support pour la nouvelle structure de l'ontologie initiale. La phase du *filtrage* va réduire

les ensembles de concepts formels des treillis finaux par l'élagage des concepts non pertinents ou superflus. Le principe consiste à : (1) préserver les concepts formels importants du domaine (les concepts de l'ontologie initiale), et (2) sélectionner les nouvelles abstractions pertinentes. La dernière étape du *rétro-codage* a pour but de générer le modèle de l'ontologie restructurée. L'idée est de traduire les concepts formels jugés pertinents dans les treillis filtrés en des éléments ontologiques sous format OWL.

Les sous-sections suivantes sont réservées à la description détaillée de chacune des phases illustrées sur un exemple simple visant à restructurer une ontologie de petite taille : CMS (Conference Management System). Tout d'abord, nous expliquons brièvement via notre exemple les types d'anomalies que notre cadre permet de résoudre.

3.4.1 Exemple de restructuration

Considérons l'ontologie CMS (la partie (a) de la figure 3.7) visualisée en UML. Elle présente un certain nombre d'anomalies par rapport à un modèle de référence (la partie (b) de la même figure). Ces anomalies sont : (1) des redondances dans les spécifications et (2) de l'incomplétude au niveau des abstractions. En effet, des classes similaires Paper et Article (même entité avec deux noms différents) sont incluses, et un même ensemble de propriétés {name, affiliation} apparait dans plusieurs classes (Author et Reviewer). L'incomplétude au niveau des abstractions est notée par l'absence, par exemple, de la classe abstraite Contributor qui devrait factoriser les propriétés qui sont partagées par les deux classes Author et Reviewer, et la classe Submission comme abstraction des deux classes Paper et Report. De plus, une analyse approfondie révèle que les relations write et compose peuvent également mener à une abstraction (relation submit) pour relier les deux nouvelles classes identifiées

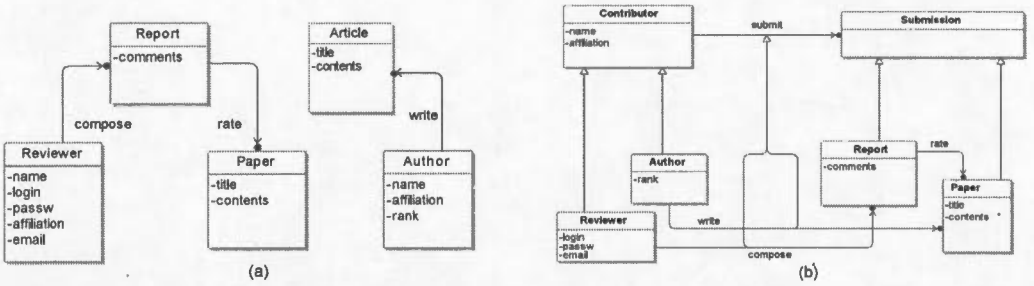


Figure 3.7 *(a) :* Fragments du modèle initial de l'ontologie CMS. *(b) :* Modèle de référence de l'ontologie CMS.

Contributor et Submission comme illustré à la figure 3.7 (b).

3.4.2 Alignement des éléments ontologiques

Les éléments ontologiques peuvent être alignés soit sur leurs noms ou bien sur leurs descriptions incluant leurs propriétés et leurs liens avec d'autres éléments (Noy et Musen, 2000; Euzenat et Valtchev, 2004). Une correspondance dans les deux aspects (le nom et la description) indique que les éléments alignés représentent la même réalité. Une correspondance dans un seul aspect pourrait signifier l'existence de problèmes de conflits de noms, à savoir des synonymes (même classe, différents noms) ou des polysémiques (même nom, différentes classes) (Dao, 2003). Ces problèmes de conflits de noms ont été traités dans le domaine de l'intégration des schémas de bases de données, et ils sont généralement résolus par la normalisation de noms basée sur des ressources linguistiques, comme des dictionnaires électroniques ou des ontologies de domaine (Rahm et Bernstein, 2001).

Notons que l'alignement est un processus complexe. Dans le cadre de notre travail, nous avons considéré un cas particulier d'alignement soit la redondance des classes avec des noms différents. L'importance de cette étape et son impact dans le processus global de restructuration peuvent être notés par deux aspects. En

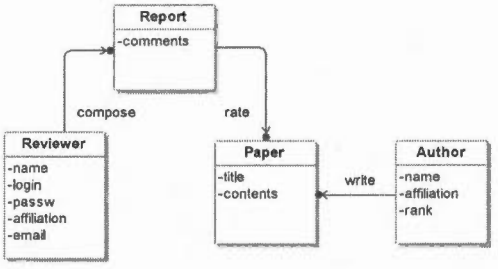


Figure 3.8 L'ontologie CMS alignée.

premier lieu, l'élimination de l'anomalie concernant les définitions redondantes de classes. Deuxièmement, l'élimination d'un facteur de bruit lié à la définition répétée des éléments similaires du modèle ontologique dans les contextes formels. Par conséquent, réduire le nombre d'abstractions à faible utilité dans les treillis finaux générés représentant l'ontologie restructurée.

Techniquement parlant, nous avons utilisé l'outil ONTOALIGN disponible dans la plateforme Inukhuk[3] (Rouane-Hacene *et al.*, 2011) pour assurer l'alignement des éléments de l'ontologie à restructurer. Cet outil inclut des APIs basées sur java : *AlignAPI*[4] et *SIMPACK*[5] pour détecter des conflits de noms potentiels dans l'ontologie initiale.

Considérons notre exemple d'ontologie à gauche de la figure 3.7, les deux classes Paper et Article représentent la même entité avec deux noms différents. L'outil d'alignement permettra de détecter ce conflit de noms et éliminera cette redondance de classes comme illustré à la figure 3.8.

---

3. Inukhuk est une plateforme orientée services pour la réingénierie d'ontologies.

4. http ://alignapi.gforge.inria.fr/

5. http ://www.ifi.uzh.ch/ddis/simpack.html

3.4.3 Codage de l'ontologie initiale

Cette phase consiste à définir les contraintes sur les entrées de notre approche de restructuration : (1) préciser le type d'ontologie visé par notre processus de restructuration ; (2) définir les étapes à suivre pour le codage de l'ontologie initiale dans la structure de données manipulée par l'ARC. Il s'agit de répondre aux deux questions suivantes :

1. *Quels sont les éléments ontologiques à coder ?*
2. *Sous quelle forme doivent être représentés les éléments ontologiques pour être traités par les méthodes de l'ARC ?*

Notons que dans le cadre de notre travail, nous visons la restructuration des ontologies *OWL*. La démarche suivie pour le codage d'une ontologie OWL dans la structure d'entrée de l'ARC est décrite dans ce qui suit.

Quels sont les éléments ontologiques à coder ?
Le codage proposé est guidé par un méta-modèle d'ontologie ODM[6], un modèle de définition d'ontologies RDF(S)/OWL.

Dans ODM, un concept ontologique réfère à une classe OWL *(owl :class)*. Une classe peut avoir des restrictions *(owl :restriction)* référant aux propriétés *(rdf :property)* dont le co-domaine peut soit être un type primitif *« datatype »* (ex. string ou double), ou bien un type complexe *« object »*, c'est-à-dire une classe. Un domaine de propriété *« domain »* est la classe dans laquelle la propriété est définie tandis que son co-domaine *« range »* est le type déclaré de sa valeur.

La figure 3.9 présente une vue simplifiée du méta-modèle ODM :OWLBase sur laquelle se base le codage.

---

6. ODM : Ontology Definition Metamodel (http ://www.omg.org/spec/ODM/1.0/).

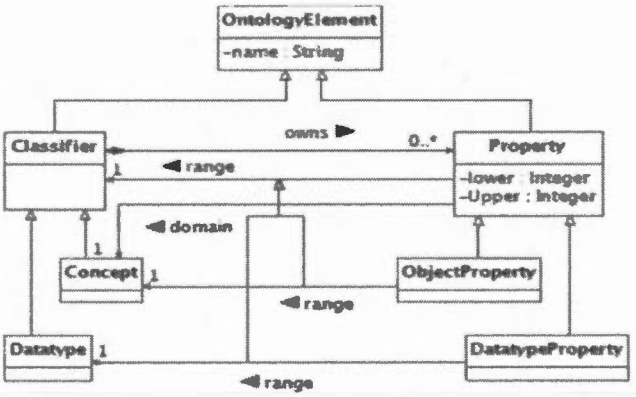


Figure 3.9 Vue simplifiée du méta-modèle ODM :OWLBase.

Les entités ontologiques codées sont : *Concept*, *ObjectProperty* et *DatatypeProperty* ainsi que les *liens inter-entités*, notamment les liens entre *Concept* et *ObjectProperty*.

Sous quelle forme doivent être représentés les éléments ontologiques pour être traités par les méthodes de l'ARC ?

L'idée est de représenter les entités ontologiques OWL par une seule FCR, il s'agit donc de définir les contextes et les relations inter-contextes.

La FCR est un format de données similaire au modèle « *entité association* ». En revanche, OWL est un langage standard pour la représentation des données sémantiques (ex., ontologie). Il est basé sur la notion de classe (inspirée des langages orientés objet) et des opérations algébriques sur les ensembles (intersection, disjonction, etc.). Visiblement, les deux langages FCR et OWL ont deux niveaux d'expressivité distincts et la transformation des données d'un format à l'autre constitue un défi de conservation d'information.

Le but de cette phase est de mettre en place un mécanisme de traduction permettant de passer des données décrites dans le langage OWL vers le format FCR.

Mécanisme de traduction

Le codage des entités ontologiques en une seule FCR suit le mécanisme suivant :

1. Chaque type d'entité *Concept* et *ObjectProperty* sera codé par un contexte formel. On notera contexte des *Classes* et contexte des *Rôles*, respectivement.
2. Les entités de type *DatatypeProperty* seront intégrées comme caractéristiques de *Concept*.
3. Les caractéristiques d'une entité dans le méta-modèle (ex., name, lower, upper, etc.) ainsi que les *DatatypeProperties* vont constituer les attributs de chaque contexte.
4. Chaque type de liens inter-entités dans ODM (ex., owns, domain, range, etc.) sera représenté par une relation binaire entre deux contextes. Ainsi, les liens entre *Concept* et *ObjectProperty* seront représentés par des relations binaires (contexte des *Classes* X contexte des *Rôles*) ou bien (contexte des *Rôles* X contexte des *Classes*).

Notons que ce mécanisme de traduction, bien qu'il soit basé sur une vue simplifiée du méta-modèle d'ontologies, permettra de coder et représenter les principaux éléments du méta-modèle OWL complet en une FCR comme proposé à l'annexe 1. Cependant, il s'agit de voir l'intérêt de représenter chaque type d'élément pour pouvoir bénéficier de l'ARC dans un but d'extraire de nouveaux liens et/ou caractéristiques. L'idée ici est de représenter uniquement les éléments utiles pour l'ARC dans notre contexte de restructuration, notamment les *concepts* et les *rôles*. Les autres éléments peuvent être ignorés au niveau du codage et à la fin du processus de restructuration seront rajoutés au niveau de l'ontologie restructurée à partir de l'ontologie initiale. Nous illustrons ce processus par le codage de l'ontologie CMS alignée, présentée à la figure 3.8, en une seule FCR. La figure 3.10 montre les deux contextes formels de cette FCR.

Classes	'author'	'reviewer'	'paper'	'report'	name	affiliation	email	rank	title	contents	comments	login	passw
Author	x				x	x		x					
Paper			x						x	x			
Reviewer		x			x	x	x					x	x
Report				x							x		

(a)

Rôles	'write'	'compose'	'rate'
write	x		
compose		x	
rate			x

(b)

Figure 3.10 Les deux contextes de *Classes* K_1 (a) et des *Rôles* K_2 (b) de l'ontologie CMS.

Notons que la propriété *« name »*, définie dans *OntologyElement* de l'ODM (voir figure 3.9), est affectée aux deux contextes de la FCR (les attributs entre ' ') et elle est utilisée pour coder les liens d'héritage entre classes (ou rôles). Dans le modèle ontologique initiale, les liens taxonomiques « is-a » existants seront traduits dans les contextes correspondants et par conséquent seront préservés dans l'ontologie restructurée. Ainsi, à chaque classe (ou rôle) est affecté non seulement son propre nom, mais également les noms de ses super-classes (ou super-rôles).

La figure 3.11 montre les relations inter-contextes constituant la FCR :

- *dom* qui représente la relation de *domaine* entre un rôle et la classe propriétaire et exprime la sémantique *« a pour domaine »*.
- *ran* qui représente la relation de *co-domaine* entre un rôle et la classe propriétaire et exprime la sémantique *« a pour co-domaine »*.
- *source* qui représente la relation *source* entre une classe et un rôle et exprime la sémantique *« est le domaine de »*.
- *target* qui représente la relation *destination* entre une classe et un rôle et exprime la sémantique *« est le co-domaine de »*.

source	Write	Compose	Rate
Author	x		
Paper			
Reviewer		x	
Report			x

target	Write	Compose	Rate
Author			
Paper	x		x
Reviewer			
Report		x	

dom	Author	Paper	Reviewer	Report
Write	x			
Compose			x	
Rate				x

ran	Author	Paper	Reviewer	Report
Write		x		
Compose				x
Rate		x		

Figure 3.11 Les relations inter-contextes de la FCR de l'ontologie CMS.

3.4.4 Analyse des données

Une fois que l'ontologie initiale est codée en une famille de contextes et de relations inter-contextes, la méthode « Multi-FCA » (décrite à la section 3.2.2) sera exécutée pour l'analyse de ces données.

Construction des treillis initiaux avec l'AFC :
La figure 3.12 montre les treillis initiaux construits à partir des deux contextes formels des *Classes* et des *Rôles* de la figures 3.10.

Le treillis des classes construit (voir à gauche de la figure) peut être interprété comme une hiérarchie de classes : les concepts sont considérés comme des classes abstraites et leurs liens d'ordre dans le treillis comme des relations taxonomiques *« is-a »*. À titre illustratif, les concepts $c_{\#4}$ et $c_{\#5}$ représentent, respectivement, les classes Author et Reviewer dans le modèle de l'ontologie alignée de la figure 3.8 . Le concept $c_{\#1}$ représente une abstraction des deux classes Author et Reviewer factorisant leurs propriétés communes à partir du contexte défini à gauche de la figure 3.10. Cette abstraction est identifiée dans le modèle de l'ontologie de référence (voir à droite de la figure 3.7) par la classe Contributor.

Notons que malgré la présence d'un nombre important de concepts dans le treillis des classes issu de l'AFC, d'autres abstractions pertinentes peuvent manquer. Par exemple, la classe Submission dans le modèle de référence n'a pas d'image au niveau

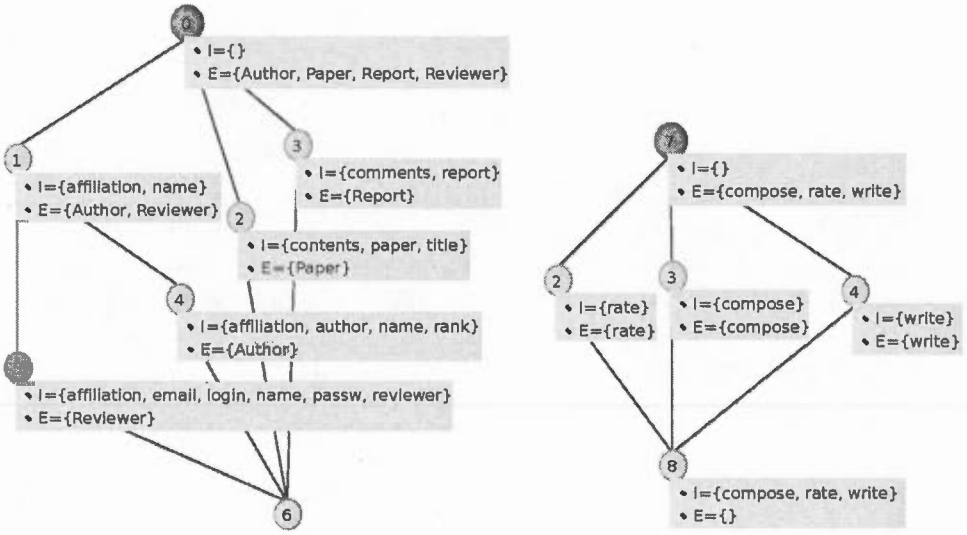


Figure 3.12 Les treillis de concepts des deux contextes K_1 (à gauche) et K_2 (à droite).

du treillis. L'utilisation de l'ARC permettra de découvrir de nouveaux éléments comme le rôle submit et la classe submission.

Traduction des relations inter-contextes en attributs binaires :

À cette étape d'analyse, le mécanisme de *graduation* est appliqué pour traduire :

- les relations inter-contextes *source* et *target* (voir figure 3.11) en utilisant le treillis des *Rôles* (voir à droite de la figure 3.12) et par conséquent étendre le contexte des *Classes* avec de nouveaux attributs relationnels.
- les relations inter-contextes *dom* et *ran* (voir figure 3.11) en utilisant le treillis des *Classes* (voir à gauche de la figure 3.12) et par conséquent étendre le contexte des *Rôles* avec de nouveaux attributs relationnels.

  Par exemple, le résultat de la *graduation* de la relation *source* (*dom)* est montré en haut (bas) de la figure 3.13.

Dans le tableau (a) illustré à la figure 3.13, les attribut *« source :$c_{\#4}$ »* et *« source :$c_{\#7}$ »* sont assignés à la classe Author puisque le rôle correspondant write (pour lequel Author

Classes	'author'	'reviewer'	'paper'	'report'	name	affiliation	email	rank	title	contents	comments	login	passw	source :c2	source :c3	source :c4	source :c7	source :c8
Author	x				x	x		x								x	x	
Paper			x						x	x								
Reviewer		x			x	x	x					x	x		x		x	
Report				x							x			x			x	

(a)

Rôles	'write'	'compose'	'rate'	dom :c0	dom :c1	dom :c2	dom :c3	dom :c4	dom :c5	dom :c6
write	x			x	x			x		
compose		x		x	x				x	
rate			x	x			x			

(b)

Figure 3.13 *(a) :* Extension du contexte des *Classes* K_1 obtenue par la graduation de la relation *« source »*. *(b) :* Extension du contexte des *Rôles* K_2 obtenue par la graduation de la relation *« dom »*.

est la source) apparait dans l'extension des concepts $c_{\#4}$ et $c_{\#7}$ dans le treillis des *Rôles* à droite de la figure 3.12, c'est-à-dire Author est relié par *source* à au moins un rôle qui est dans l'extension de ces concepts. Dans le tableau (b) de la figure, les attributs « dom :$c_{\#0}$ », « dom :$c_{\#1}$ » et « dom :$c_{\#4}$ » sont assignés au rôle write puisque la classe domaine correspondante Author apparait dans l'extension des concepts $c_{\#0}$, $c_{\#1}$ et $c_{\#4}$ dans le treillis des *Classes* à gauche de la figure 3.12, c'est-à-dire write est relié par dom à au moins une classe qui est dans l'extension de ces concepts.

Construction des treillis finaux :

Cette dernière étape de l'analyse consiste à générer les treillis finaux. Ces treillis finaux représentent la nouvelle réorganisation des éléments de l'ontologie initiale. La figure 3.14 montre le point fixe des treillis finaux associés aux deux contextes des *Classes* (a) et des *Rôles* (b) de l'ontologie CMS. Notons que les *intensions* et

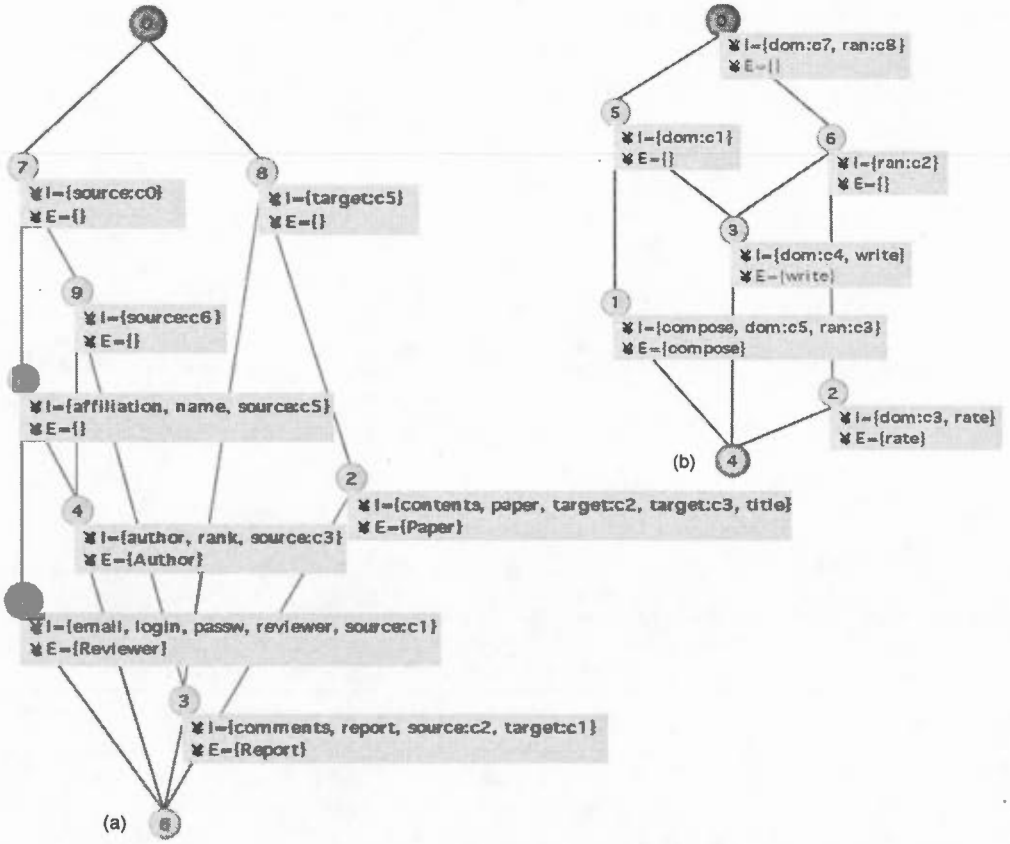


Figure 3.14 Le point fixe des treillis finaux de l'ontologie CMS. *(a) :* Treillis final du contexte des *Classes*. *(b) :* Treillis final du contexte des *Rôles*.

les *extensions* des concepts sont réduits pour des raisons de clarté. En effet, un objet (ou attribut) est uniquement montré dans le noeud du concept minimal (ou maximal) que l'extension (ou intension) caractérise. L'héritage ascendant (ou descendant) entre concepts est utilisé pour retrouver leurs extensions (ou intensions) complètes.

Rappelons que dans les treillis finaux, un attribut relationnel *« r :c »* est interprété comme étant une association entre deux concepts. Le premier concept est celui qui contient *« r :c »* dans son ensemble *intension* et le deuxième concept est celui auquel l'attribut réfère explicitement (le concept c). Par exemple, dans le treillis des *Classes* à gauche de la figure 3.14, l'attribut « source :$c_{\#5}$ » du concept $c_{\#1}$ dans le treillis des *Classes* est interprété comme suit : *Pour chaque classe* o *dans l'extension de* $c_{\#1}$ {*Author, Reviewer*}, *il existe*[7] *un rôle* p *dans l'extension de* $c_{\#5}$ {*compose, write*} *(dans le treillis des Rôles) tel que* o *est la source de* p.

D'autre part, dans le treillis des *Rôles* (à droite de la figure 3.14), le concept $c_{\#5}$ résume les liaisons communes entre write et compose qui correspondent à une classe *domaine* partagée représentée par le concept $c_{\#1}$ dans le treillis des *Classes* qui est la super-classe de Author et Reviewer.

Notons que les treillis finaux constituent le support pour la nouvelle réorganisation de l'ontologie initiale, c'est-à-dire l'ontologie restructurée. Cependant, l'interprétation directe de ces treillis comme une hiérarchie de classes mène à un modèle ontologique complexe avec des concepts fortuits. Pour remédier à ce problème et par conséquent construire une hiérarchie plus compacte, nous proposons de faire un filtrage de ces treillis pour élaguer tous les concepts non pertinents.

---

7. Dans notre exemple, les attributs relationnels réfèrent à un quantificateur existentiel.

3.4.5 Filtrage des treillis de concepts

Bien que l'AFC nous fournisse un cadre complet et exhaustif de réorganisation, elle nous pose un sérieux problème, celui de la complexité et la taille des treillis générés, dépendamment du nombre d'éléments du modèle de l'ontologie initiale, qui peuvent contenir beaucoup de concepts superflus. L'ARC tend à générer des treillis plus larges du fait qu'elle traite des structures plus riches (FCR).
Dans notre contexte, un *concept formel* représentera une *classe* (ou concept ontologique) dont l'intension et l'extension correspondent, respectivement, à l'ensemble des propriétés et l'ensemble des sous-classes.

Dans le treillis final produit par l'ARC, chaque concept formel est un candidat susceptible d'être sélectionné comme une classe dans l'ontologie restructurée si jugée *pertinent* (ou non si *non pertinent*). À ce stade, des décisions importantes sont à prendre en réponse à deux questions essentielles :

- *Quelle est la valeur ontologique d'un concept formel ?*
- *Comment reconnaire parmi le grand nombre de concepts formels candidats ceux qui sont pertinents ?*

Dans les travaux existants, notamment ceux qui traitent de l'apprentissage et la fusion d'ontologies en utilisant l'AFC, citons (Stumme et Maedche, 2001; Nanda *et al.*, 2006), il n'existe pas de suggestions sur une méthode d'évaluation automatique de la qualité d'un concept formel. La majorité des travaux propose une validation basée sur les experts pour la génération de l'ontologie cible à partir des treillis finaux de concepts formels. Nous avons tenté d'adresser cette issue dans notre contexte de restructuraion par l'ARC. Notre objectif est de proposer des mesures pour évaluer la pertinence des concepts formels qui vont constituer l'ontologie restructurée.

Pour cela, nous nous sommes inspirés : (1) des principes et des exigences qui

doivent être respectés par un modèle ontologique (Gómez-Pérez, 1999) définis à la section 2.1.3 ; (2) les travaux sur l'évaluation des ontologies, incluant ceux qui considèrent une ontologie comme un graphe et tentent de détecter ses caractéristiques structurelles et sémantiques (Alani et Brewster, 2006) ; et (3) des mesures pour le filtrage des treillis (Kuznetsov *et al.*, 2007; Roth *et al.*, 2008). Nous avons choisi quatre mesures dont deux sont structurelles : *densité* qui indique l'utilité d'un concept en terme d'information additionnelle qu'il apporte par rapport à son voisinage, et *stabilité* qui mesure l'insensibilité de la description d'un concept aux bruits, c'est-à-dire la probabilité pour un concept de préserver son intension même s'il perd un certain nombre d'objets de son extension. Les deux autres mesures sont basées sur la sémantique : *la proximité sémantique avec le centre d'intérêt de l'utilisateur* qui évalue le niveau d'encrage, en termes de liens directs ou indirects, d'un nouveau concept formel avec les concepts représentant les classes de l'ontologie initiale, et *la proximité sémantique entre les concepts enfants* qui tente d'indiquer si un concept est correcte sémantiquement, c'est-à-dire mesurer jusqu'à quel point l'extension subsume des sous-concepts similaires.

Le filtrage que nous proposons consiste à appliquer ces mesures sur les trellis finaux représentant la nouvelle restructuration de l'ontologie pour élaguer les concepts superflus (non pertinents). La figure 3.15 montre le résultat de l'application de la méthode de filtrage sur notre exemple d'ontologie CMS :

1. Les concepts $c_{\#0}$, $c_{\#7}$ et $c_{\#6}$ du treillis des *Classes* ont été jugés non pertinents par la méthode et donc élagués. En effet, ces concepts n'ont pas d'images dans le modèle de référence montré à droite de la figure 3.7.

2. Les concepts $c_{\#1}$, $c_{\#2}$, $c_{\#3}$, $c_{\#4}$, $c_{\#5}$, $c_{\#8}$ et $c_{\#9}$ ont été jugés comme pertinents et donc filtrés. Parmi ces concepts :
   - Les concepts $c_{\#1}$, $c_{\#2}$, $c_{\#3}$, $c_{\#4}$, $c_{\#5}$ et $c_{\#8}$ ont bien des images dans l'ontologie de référence et représentent, respectivement, les classes Contributor,

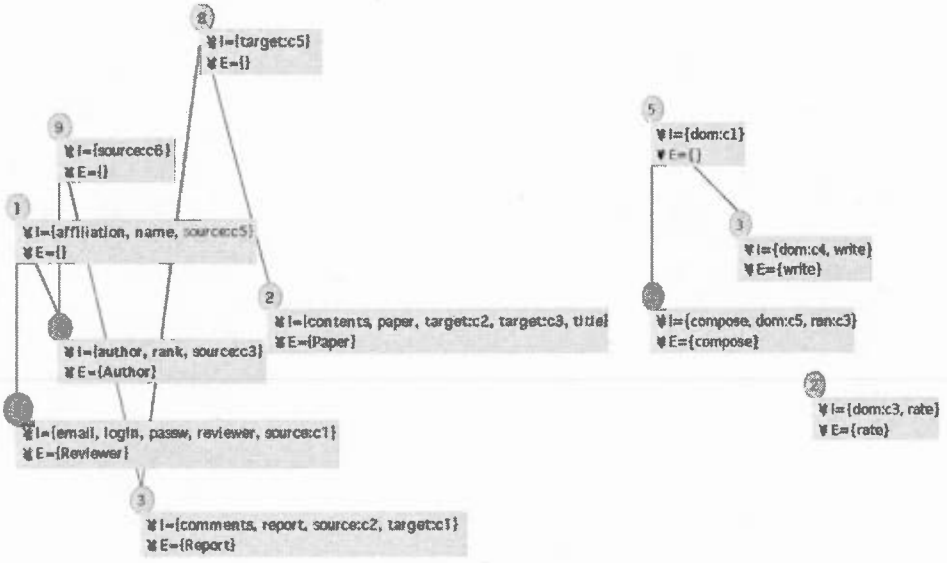


Figure 3.15 Les deux ensembles de concepts formels filtrés.

Paper, Report, Author, Reviewer et Submission.

– Le concept $c_{\#9}$ n'a pas son équivalent dans l'ontologie de référence.

En résumé, la méthode de filtrage aura en entrée *la FTR (ou les treillis finaux)* et retourne en sortie *une famille d'ensembles de concepts filtrés*. Le chapitre suivant détaille les mesures retenues ainsi que la méthode de filtrage proposée.

3.4.6 Rétro-codage de l'ontologie restructurée

Le rétro-codage a pour but de traduire la famille des ensembles de concepts formels filtrés à partir des deux treillis de concepts *Classes* et *Rôles* en un modèle ontologique OWL restructuré. Notons ici l'avantage d'utiliser l'ARC qui offre une interprétation facile et par assemblage des deux treillis directement en une ontologie consistante. Cependant, l'étape de traduction est délicate du fait de la nature transitive et potentiellement circulaire des dépendances entre les abstractions qui

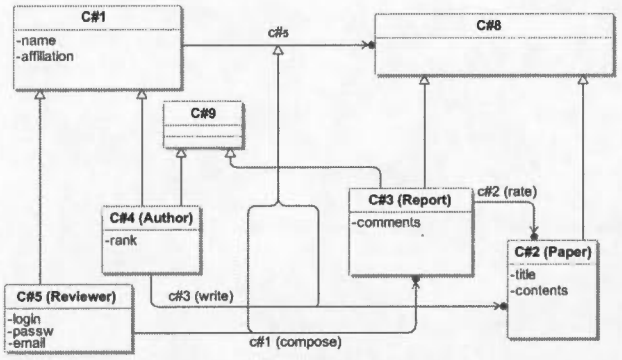


Figure 3.16 Le modèle de l'ontologie CMS restructuréé.

sont induites par des relations dans les intensions des concepts filtrés. La méthode de rétro-codage développée se base sur une méthode exploratoire qui implique des outils de navigation de treillis ainsi que l'expert de l'ontologie.

Le mécanisme de traduction des concepts formels filtrés en des éléments ontologiques suit les règles suivantes :

- Chaque concept formel filtré du treillis des *Classes* est traduit en une classe OWL.
- L'ensemble des attributs formels fournis par les intensions exprimant des informations du domaine (propriétés des classes dans l'ontologie initiale) sont traduits comme des *« datatype »* au niveau de la classe correspondante.
- Les liens d'héritage *« is-a »* entre les classes (ou object properties) sont déduits à partir de la relation d'ordre entre les concepts filtrés du treillis des *Classes* (ou du treillis des *Rôles*).
- Enfin, l'ensemble des concepts filtrés du treillis des *Rôles* est visité afin de déterminer la classe *domaine* et la classe *co-domaine* de chaque *object property*[8].

8. Pour plus de détails, voir l'algorithme 1 défini à la section 4.3.2. L'algorithme montre comment les deux treillis des *Classes* et des *Rôles* sont visités itérativement suivant des liens inter-concepts établis, en particulier, par les attributs relationnels de type *« dom :c »* et *« ran :c »*

Le résultat de la traduction est un fichier OWL. La figure 3.16 illustre le modèle restructuré de l'ontologie CMS généré à partir des deux treillis filtrés de la figure 3.15. En comparant ce dernier avec son modèle de référence de la figure 3.7 (à droite), nous pouvons faire les observations suivantes :

- La restructuration globale de l'ontologie est très proche du modèle de référence.
- Les anomalies existantes dans le modèle initial de l'ontologie de la figure 3.7 (à gauche) et discutées à la section 3.4.1 ont été corrigées :
  - La classe redondante Article a été supprimée.
  - L'ensemble des propriétés communes {name, affiliation} entre les deux classes Author et Reviewer ont été déplacées vers une nouvelle classe Contributor qui est rajoutée comme une abstraction de ces deux classes.
  - La classe Submission qui manquait dans l'ontologie initiale a été également rajoutée.
  - Un nouveau rôle submit ayant comme domaine la classe Contributor et comme co-domaine la classe Submission a été rajouté comme une abstraction des deux rôles write et compose.

Notons que :

1. Contrairement aux classes issues des concepts $c_{\#1}$ et $c_{\#8}$ (ou le rôle issu de $c_{\#5}$), les classes de $c_{\#2}$, $c_{\#3}$, $c_{\#4}$ et $c_{\#5}$ (ou les rôles de $c_{\#1}$, $c_{\#2}$ et $c_{\#3}$) qui existaient dans l'ontologie initiale sont nommées automatiquement par leurs noms respectifs : Paper, Report, Author, Reviewer (ou compose, rate et write).
2. Étant donné que le filtrage n'est pas parfait, la classe $c_{\#9}$ représente une abstraction erronée qui a été gardée alors qu'elle n'a pas de correspondance dans le modèle de référence.

Finalement, le nouveau modèle de l'ontologie restructuré sera présenté à l'expert du domaine pour une validation finale. Cette tâche de validation peut être facili-

---

afin d'identifier le concept *domaine* et le concept *co-domaine* de chaque *rôle*.

tée par le nommage automatique des nouvelles abstractions (attribution de noms significatifs). Une alternative consiste à composer les noms des entités nommées qui mènent à la création de l'entité abstraite. Une autre alternative serait de trouver une interprétation sémantique dans une ressource externe comme Wordnet [9], Wikipedia [10], etc. L'idée est de trouver, dans ces ressources, un terme généralisant les noms des éléments abstraits.

En résumé, le cadre global de la restructuration décrit précédemment permet de traiter certaines des anomalies ou imperfections qui peuvent être détectées dans un modèle ontologique. Cependant, il possède une limite liée au fait qu'il ne traite pas toutes les anomalies, notamment, tous les cas d'incomplétude. La section suivante discute cet aspect.

3.5 Traitement des anomalies par l'approche de restructuration

Cette section résume certaines des anomalies citées à la section 2.1.6 qui sont traitées par notre cadre. Un exemple de restructuration dans le cas de chaque anomalie est donné pour illustrer tout d'abord comment cette erreur se présente au niveau de l'ontologie et ensuite comment elle est corrigée par notre approche.

3.5.1 Anomalies traitées par le cadre conceptuel de l'approche

Nous avons pu montré dans la section précédente, via un exemple simple d'ontologie, comment un certain nombre d'anomalies sont traitées automatiquement par le cadre conceptuel de l'approche. En effet, ces anomalies seront corrigées au niveau des treillis finaux (représentant la nouvelle réorganisation de l'ontologie de

9. http ://wordnet.princeton.edu/

10. http ://www.wikipedia.org/

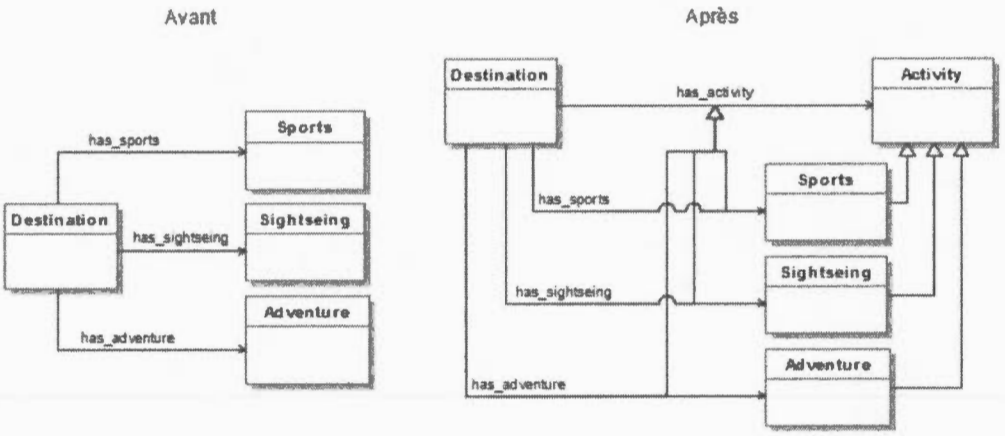


Figure 3.17 Exemple de restructuration dans le cas *d'incomplétude.*

départ) avant la phase du filtrage. Citons :

- *Incomplétude :* Une partie des connaissances manquantes sera détectée (par abstraction) à partir des redondances au niveaux des spécifications des éléments de départ (voir les exemples des deux figures 3.17 et 3.18).
- *Redondance dans les affectations des propriétés :* L'ensemble des propriétés répétées dans plusieurs classes seront déplacées vers la super-classe si cette dernière existe ou bien une nouvelle abstraction, qui factorisera les propriétés communes, sera créée (voir l'exemple de la figure 3.18).
- *Redondance de classes :* La répétition de classes similaires (ayant la même définition formelle) sera éliminée lors de la phase *d'alignement* (voir l'exemple de la figure 3.19).
- *Redondance de relations taxonomiques :* Les relations *« is-a »* redondantes entre classes sont éliminées automatiquement au niveau du treillis. En effet, le treillis ignorera les liens de spécialisation transitifs (voir l'exemple de la figure 3.20).

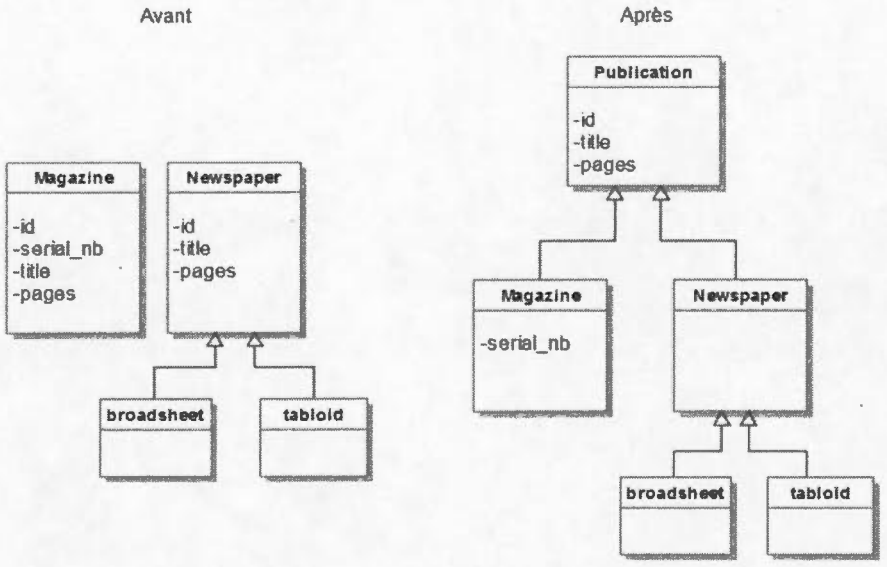


Figure 3.18 Exemple de restructuration dans le cas de *redondance dans les affectations des propriétés* et *d'incomplétude.*

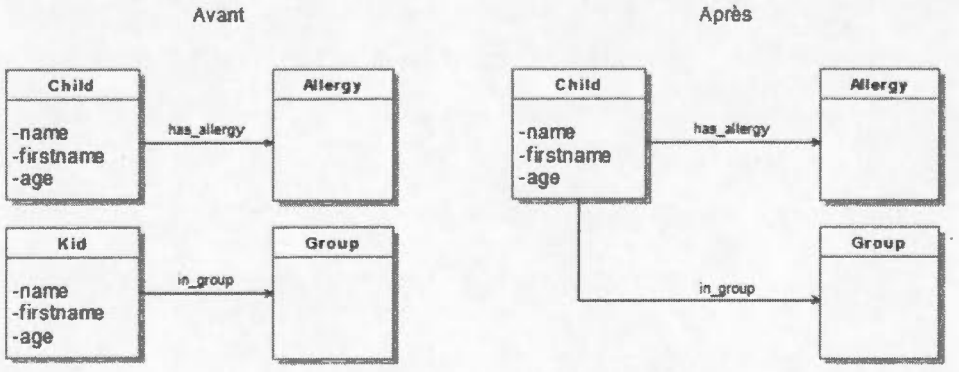


Figure 3.19 Exemple de restructuration dans le cas de *redondance de classes.*

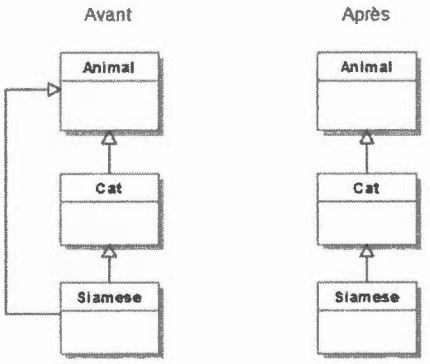


Figure 3.20 Exemple de restructuration dans le cas de *redondance de relations « is-a »*.

3.5.2 Anomalies qui pourraient être traitées par les mesures

D'autres types d'erreurs telles que *chaines d'héritage* et *erreurs sémantiques* ne sont pas corrigées automatiquement par le cadre. Cependant, elles peuvent être traitées au niveau du filtrage comme cela sera expliqué dans le chapitre suivant.

Par exemple, les classes cl_2, cl_3,..., cl_{n-1} appartenant à une chaine d'héritage cl_1 *is-a* cl_2...*is-a* cl_n pourraient être éliminées pour leurs faibles valeurs par les mesures évaluant la pertinence structurelle de leurs concepts images au niveau du treillis (voir l'exemple de la figure 3.21) conformément à la section 4.3.1. De même, les classes qui regroupent des sous-classes non similaires pourraient être éliminées ou corrigées par les mesures évaluant la pertinence sémantique de leurs concepts images au niveau du treillis (voir l'exemple de la figure 3.22) conformément à la section 4.3.4.

Dans ce qui suit, nous citons les erreurs non traitées :

- *Erreurs de circularité*
- *Erreurs de partitions*
- *Manque d'informations nécessaires sur classes/propriétés/domaines/co-domaines*

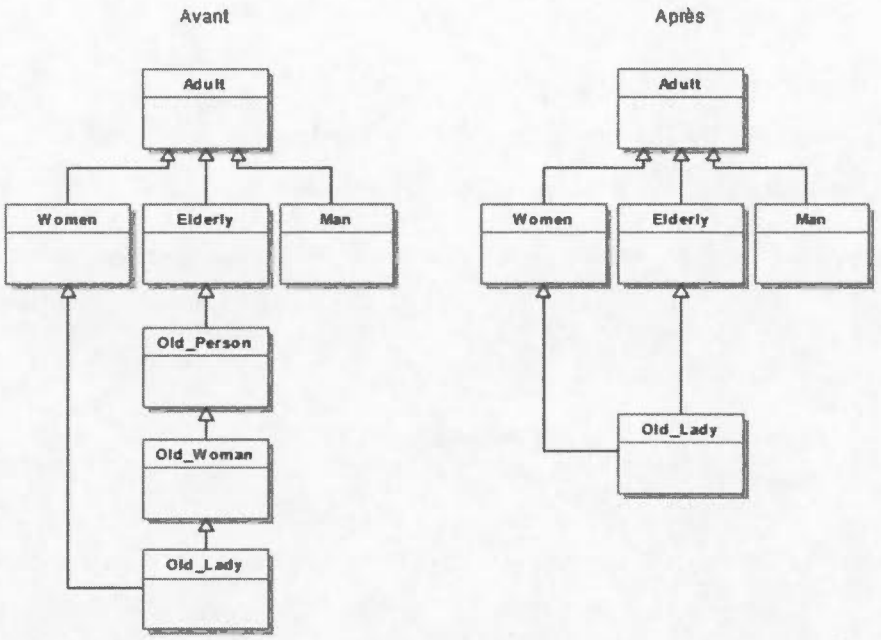


Figure 3.21 Exemple de restructuration dans le cas de *chaine d'héritage.*

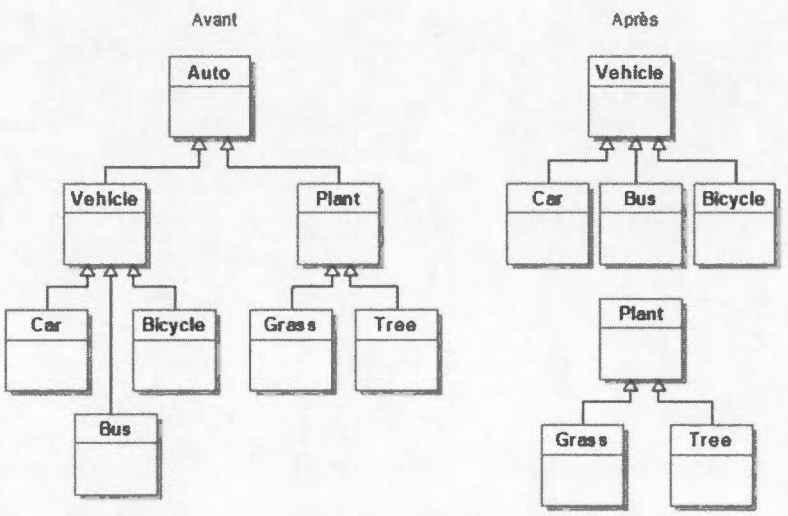


Figure 3.22 Exemple de restructuration dans le cas *d'erreur sémantique.*

- *Classe/propriété inutile*
- *Classe disjointe isolée*
- *Domaine/co-domaine trop spécifique*

En résumé, la contribution de notre approche au domaine de la restructuration des modèles ontologiques est notée par les points suivants :

1. L'approche se base sur un cadre formel qui assure une restructuration globale, complète et exhaustive des données contenues dans le modèle initial.
2. Elle est complètement automatique.
3. Elle fournit une structure hiérarchique des *« propriétés objet »*.
4. Elle est généralisable et ne dépend pas d'un contexte spécifique.

Enfin, la comparaison de notre approche aux méthodes de restructuration étudiées est montrée par la dernière ligne du tableau 3.2 déjà défini à la section 2.2.4 (tableau 2.3).

Approche	Anomalies traitées					Démarche de restructuration			
	Incons.	Incomp.	Red.	Défic.	Autres	Type	Formalisat.	Automatisat.	Généralisat.
(Gailly et Poels, 2007)	✓	✓	?	-	✓	RL	NF	M	S
(Kehagias *et al.*, 2008; Kehagias *et al.*, 2010)	✓	-	✓	✓	✓	RL	F	M	S–»G
(Wimmer, 2009)	?	✓	?	✓	✓	RL	NF	SA	G
(Suarez-Figueroa *et al.*, 2012)	?	✓	?	-	✓	RL	NF	M	S–»G
(Costa *et al.*, 2013)	✓	✓	?	-	✓	RG	F	A	S
(Šváb-Zamazal *et al.*, 2008)	✓	-	-	-	✓	RL	NF	M	G
(Behkamal *et al.*, 2010)	✓	-	-	-	✓	RL	NF	M	G
(Törniai *et al.*, 2013)	✓	-	✓	✓	✓	RL	NF	M	S–»G
(Baumeister et Seipel, 2006)	✓	-	✓	✓	-	RL	NF	M	G
(Ostrowski, 2008)	-	-	-	-	✓	RL	NF	A	G
(Gröner et Staab, 2010)	✓	-	?	✓	✓	RL	F	M	G
Notre approche	-	✓	✓	✓	-	RG	F	A	G

Tableau 3.2 Comparaison de l'approche proposée aux méthodes de restructuration étudiées.

3.6 Conclusion

Tout au long de ce chapitre, nous avons expliqué le cadre développé pour la restructuration des ontologies par l'ARC. Nous avons pu montrer, via un exemple illustratif, que l'ARC constitue un cadre formel très approprié pour la restructuration, notamment pour l'identification des connaissances manquantes.

Une phase cruciale du processus global, soit *le filtrage*, a révélé le besoin d'un moyen pour le filtrage des concepts formels potentiellement pertinents, parmi un grand nombre de candidats pouvant constituer les concepts de l'ontologie restructurée. On peut admettre qu'un bon processus de filtrage améliorera considérablement le résultat du processus global de restructuration et pourrait aboutir à un enrichissement plus efficace de l'ontologie initiale.

Le chapitre suivant est consacré au développement détaillé de la méthode que nous proposons comme solution à la problématique du filtrage des treillis de concepts issus de l'ARC, en d'autres termes, notre proposition pour la prise en charge de la *phase de filtrage* du cadre de restructuration décrit dans ce chapitre.

CHAPITRE IV

MÉTHODE POUR LE FILTRAGE DES TREILLIS DE CONCEPTS

Ce chapitre constitue la contribution principale de notre travail. Il sera consacré au traitement de la problématique du filtrage des treillis de concepts dans le contexte de restructuration exposé dans le chapitre précédent. Nous proposons une méthode basée essentiellement sur des mesures d'évaluation de la pertinence des concepts formels pour en retenir ceux qui constitueront les concepts du modèle ontologique restructuré.

La pertinence étant une propriété contextuelle et subjective, la sélection des concepts pertinents dans un treillis constitue une tâche délicate pour laquelle peu de solutions sont disponibles dans la littérature. Les méthodes de filtrage complètement automatiques se basent sur des propriétés structurelles qui sont faciles à mesurer (Kuznetsov *et al.*, 2007). Cependant, dans notre contexte, nous comptons approximer cette pertinence en alliant les niveaux, *structurel* et *sémantique*. Au niveau sémantique, nous considérons que l'ontologie initiale, bien que sa structure soit imparfaite, constitue une source riche de connaissances du domaine à explorer dans la conception des mesures d'évaluation. Elle constitue donc un facteur important de la *pertinence sémantique*.

Dans ce chapitre, nous formulons tout d'abord la problématique du filtrage et

nous définissons un profil pour un concept formel pertinent. Nous abordons ensuite les mesures que nous avons retenues pour l'évaluation de la pertinence de chaque concept formel au niveau du treillis. Pour chaque mesure, nous précisons la motivation derrière son choix, sa description et sa formulation ainsi que son interprétation dans le contexte ontologique. Enfin, la structure globale de l'algorithme de filtrage est présentée en donnant un certain nombre d'heuristiques de filtrage formalisant des règles d'application sur les mesures développées.

4.1 Problématique du filtrage

Dans notre cadre de restructuration, le problème de filtrage se pose principalement pour le treillis des *Classes*. Nous le définissons comme suit :
Ayant un treillis relationnel des *Classes*, $T_{classes}$, le problème de filtrage consiste à obtenir un ensemble de concepts formels, $CF_{filtres}$, tels que :

1. Les concepts formels dans $CF_{filtres}$ sont inclus dans l'ensemble des concepts du treillis $T_{classes}$:
$$\forall cf(cf \in CF_{filtres} \longrightarrow cf \in T_{classes})$$
2. $CF_{filtres}$ inclut les concepts du domaine (CD) non erronés. Les concepts formels dans $T_{classes}$ représentant les classes de l'ontologie initiale non erronées [1] doivent être filtrés (ne doivent pas être élagués) :
$$\forall cf(cf \in CD \wedge NonErrone(cf) \longrightarrow cf \in CF_{filtres})$$
3. Si cf_1 et cf_2 sont deux concepts formels dans $CF_{filtres}$ et s'il existe une relation (directe ou indirecte) de généralisation entre eux dans $T_{classes}$, alors cette relation doit aussi exister entre eux dans $CF_{filtres}$:

---

1. Les classes qui sont considérées comme erronées et peuvent être élaguées sont, en particulier, *les classes redondantes, les classes non correctes sémantiquement* et *les classes appartenant à une chaines d'héritage.*

$$\forall cf_1, cf_2 (cf_1 \in CF_{filtres} \wedge cf_2 \in CF_{filtres} \wedge relIsa(cf_1, cf_2) \in T_{classes} \longrightarrow relIsa(cf_1, cf_2) \in CF_{filtres})$$

4. Tout concept dans $CF_{filtres}$ qui n'est pas un concept de domaine représente une nouvelle abstraction découverte qui est probablement pertinente :

$$\forall cf (cf \in CF_{filtres} \wedge cf \notin CD \longrightarrow cf \textit{ est probablement pertinent})$$

Dans le but de trouver une solution à cette problématique, notre idée de base est de chercher un moyen pour évaluer *« la pertinence »* de chaque concept formel en prenant en considération les deux points suivants : *(1) Structures de données riches (FCR)* ; et *(2) Contexte ontologique.*

Plusieurs définitions ont été données à la notion de pertinence, notamment en Recherche d'information (Schamber *et al.*, 1990; Borlund, 2003). Nous retenons celle de (Schamber *et al.*, 1990) : *« a multidimensional concept, that is dependent on both internal (cognitive) and external (situational) factors, that is based on a dynamic human judgment process, and that is a complex but systematic and measurable phenomenon ».*

La démarche que nous avons adoptée pour évaluer la pertinence d'un concept formel est la suivante :

1. Définir un profil pour un concept formel pertinent, c'est-à-dire déterminer un certain nombre de critères auxquels doit répondre un concept formel pour qu'il soit accepté comme un concept ontologique pertinent.
2. Traduire, dans la mesure du possible, chaque critère en une mesure permettant son évaluation.
3. Proposer des heuristiques basées sur les métriques retenues pour évaluer la pertinence de chaque concept formel.
4. Proposer un algorithme pour le filtrage des treillis ; un algorithme qui mettra en œuvre les notions présentées en 1, 2 et 3.

4.2 Profil d'un concept formel pertinent

Définir le profil exact d'un concept formel *pertinent*, notamment dans un contexte ontologique, est une tâche difficile. Nous avons, dans le cadre de notre travail, défini un certain nombre de critères pour guider le choix de mesures à explorer pour estimer la pertinence.

Dans ce but, nous nous sommes posés la question sur quoi nous pouvons raisonner. La réponse est de considérer les deux aspects d'un modèle ontologique. Le premier est la syntaxe, c'est-à-dire *la structure* ou la topologie du graphe, il s'agit de vérifier si la structure a été améliorée. L'autre aspect est la pertinence sur le plan *sémantique*, ce qui est délicat et plusieurs possibilités existent. Nous avons choisi une alternative dont l'évaluation semble être la plus immédiate.

Nous proposons donc deux catégories de critères selon le type de pertinence à évaluer : les critères d'évaluation de ***la pertinence structurelle*** et les critères d'évaluation de ***la pertinence sémantique***.

Afin de définir un ensemble de critères auxquels doit répondre un concept formel pour qu'il soit accepté comme un concept ontologique, nous nous sommes inspirés premièrement des exigences ou des principes que doit respecter un modèle ontologique (Gómez-Pérez, 1999) décrits à la section 2.1.3. Nous avons retenu un des critères qui concerne particulièrement les concepts et qui pourrait être utile pour l'évaluation de la pertinence, soit, *la distance sémantique minimale entre les concepts enfants*. Ensuite, nous avons considéré quelques travaux sur l'évaluation des ontologies, notamment ceux qui considèrent une ontologie comme un graphe et tentent de détecter ses caractéristiques structurelles et sémantique (Rada *et al.*, 1989; Poesio et Almuhareb, 2005; Alani et Brewster, 2006). L'analyse de ces travaux nous ont mené à opter pour les deux critères suivants : (1) *riche description et forte interconnexion*, et (2) *proximité sémantique avec le centre d'intérêt de l'uti-*

lisateur. Par ailleurs, des travaux sur le filtrage des treillis de concepts (Kuznetsov *et al.*, 2007; Roth *et al.*, 2008) nous ont inspiré. Nous nous sommes intéressés notamment à *la stabilité d'un concept*, c'est-à-dire l'insensibilité de la description d'un concept aux bruits.

En résumé, les critères sélectionnés sont regroupés comme suit selon le type de pertinence à évaluer.

Critères d'évaluation de la pertinence structurelle

L'évaluation de la pertinence structurelle d'un concept formel consiste à chercher à quel point ce concept est important dans la structure actuelle du treillis. Les critères qui peuvent couvrir ce type de pertinence sont :

- *Critère 1 :* Stabilité de la description (insensibilité aux bruits)
- *Critère 2 :* Riche description et forte interconnexion au sein du graphe ontologique

Critères d'évaluation de la pertinence sémantique

L'évaluation de la pertinence sémantique d'un concept formel consiste, d'une part, à indiquer à quel point ce concept est correct sémantiquement (regroupe des sous-concepts similaires) et, d'autre part, à chercher à quel point il est proche sémantiquement du centre d'intérêt de l'utilisateur. Les critères qui peuvent couvrir ce type de pertinence sont :

- *Critère 3 :* Proximité sémantique entre les concepts enfants
- *Critère 4 :* Proximité sémantique avec le centre d'intérêt de l'utilisateur

Nous discutons dans la section suivante un certain nombre de mesures que nous avons retenues et/ou adaptées pour mesurer les critères définis ci-dessus. Pour chaque mesure, nous mentionnons en premier lieu la motivation derrière le choix du critère que la mesure tentera d'évaluer. Ensuite, nous donnons sa description et sa formule de calcul ainsi que le pseudo-code de la méthode guidant les règles de son application. Enfin, nous projetons cette mesure dans notre contexte de

restructuration d'ontologies et proposons une interprétation de ses résultats.

4.3 Mesures d'évaluation de la pertinence

4.3.1 Mesure de stabilité

Motivation

Il est bien plausible d'inclure ou d'exclure un concept de domaine vis-à-vis des engagements faits par les concepteurs sur l'ontologie construite. Cependant, il est bien accepté que toute classe/abstraction admise dans l'ontologie ne devrait pas dépendre de l'inclusion ou l'exclusion d'autres concepts du domaine. Sinon, cette classe serait instable, c'est-à-dire très sensible aux changements.

Ce critère est le premier que nous avons retenu pour estimer la pertinence d'un concept dans l'ontologie restructurée. Pour cela, nous avons choisi d'appliquer la notion de *stabilité* pour approximer cette pertinence.

Description et formulation

L'idée de la stabilité a été utilisée pour estimer la plausibilité des hypothèses de différents types. Dans cette ligne de réflexion, (Kuznetsov, 2007) a introduit la réalisation de l'idée de la stabilité des hypothèses basée sur la similarité des descriptions d'objets, et l'a étendu aux concepts formels. Suivant cette même idée, nous avons étudié l'utilité de cette mesure pour l'évaluation de la pertinence des concepts formels en tant que concepts ontologiques. Ici, nous utilisons la définition de stabilité comme suit (Kuznetsov, 2007) :

Définition 8. Soient $K=(O,A,I)$ un contexte formel et (X,Y) un concept formel de K. *L'indice de stabilité*, σ, de (X,Y) est défini par :

$$\sigma(X,Y) = \frac{|\{Z \subseteq X | Z' = X' = Y\}|}{2^{|X|}} \tag{4.1}$$

```
Algorithm ComputeStability
  Concepts := 𝔅(𝕂)
  for each (A, B) in Concepts
    Count[(A, B)] := the number of lower neighbors of (A, B)
    Subsets[(A, B)] := 2^|A|
  end for
  while Concepts is not empty
    let (C, D) be any concept from Concepts with Count[(C, D)] = 0
    Stability[(C, D)] := Subsets[(C, D)] / 2^|C|
    remove (C, D) from Concepts
    for each (A, B) > (C, D)
      Subsets[(A, B)] := Subsets[(A, B)] − Subsets[(C, D)]
      if (A, B) ≻ (C, D)
        Count[(A, B)] := Count[(A, B)] − 1
      end if
    end for
  end while
  return Stability
```

Figure 4.1 Algorithme de calcul de la stabilité (Roth *et al.*, 2008).

L'indice de stabilité d'un concept, $\sigma(X, Y)$, indique à quel point l'intension d'un concept dépend d'objets particuliers de l'extension. En d'autres termes, l'indice de stabilité représente la probabilité pour un concept de préserver son intension même s'il perd un certain nombre d'objets de son extension. L'idée derrière la stabilité est que *« une intension stable est probablement "réel" même si la description de certains objets est "bruitée" »*.

Les auteurs dans (Roth *et al.*, 2008) ont introduit un algorithme simple illustré à la figure 4.1, que nous avons appliqué ici, qui considère le graphe de couverture d'un treillis de concepts $\mathcal{B}(K)$ et calcule l'indice de stabilité pour chaque concept du treillis. Ci-après, nous donnerons une brève explication de cet algorithme.

Interprétation de l'algorithme

L'algorithme (voir figure 4.1) traverse le graphe de couverture et calcule les indices de stabilité de tous les concepts du treillis du bas (concept *bottom*) vers le haut

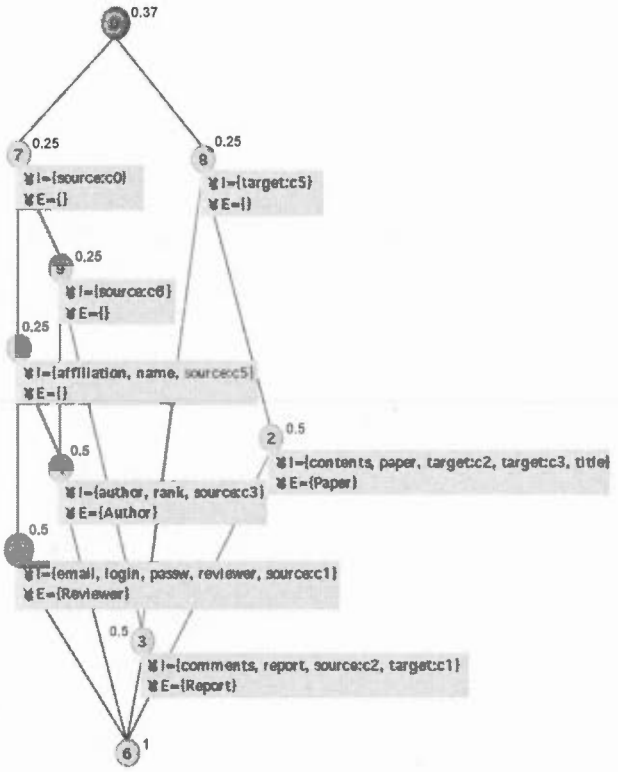


Figure 4.2 Valeurs de stabilité des concepts formels du treillis des *Classes*.

(concept *top*). Un concept n'est traité qu'une fois les indices de stabilité de tous ses sous-concepts ont été calculés.

Pour déterminer l'indice de stabilité $\sigma(A, B)$, en premier le nombre de sous-ensembles $E \subseteq A$ qui génèrent l'intersection $B(E' = B)$ est calculé, et ensuite ces sous-ensembles seront sauvegardés dans *Subsets*. $\sigma(A, B)$ est le nombre de ces sous-ensembles divisé par le nombre de tous les sous-ensembles de A, soit, par $2^{|A|}$. Une fois calculé, $\sigma(A, B)$ est sauvegardé dans *Stability*, qui est retourné par l'algorithme.

Exemple :

La figure 4.2 montre le treillis final des *Classes* de l'ontologie CMS[2]. Chaque concept formel lui est étiquetée la valeur de son indice de stabilité. Pour le concept $c_{\#7}$, par exemple :

$c_{\#7}$=(A,B)=({Reviewer, Author, Report}, {source :c0})

$2^{|A|} = 2^3 = 8$, nous avons donc 8 sous-ensembles de A dont les *intensions* respectives sont :

$\emptyset'$= {comments, report, source :c2, target :c1, email, login, passw, reviewer, source :c1, author, rank, source :c3, contents, paper, target :c2, target :c3, title, name, affiliation, source :c5, source :c6, target :c5, source :c0} $\neq$ {source :c0}

{Reviewer}$'$={email, login, passw, reviewer, source :c1, name, affiliation, source :c5, source :c0} $\neq$ {source :c0}

{Author}$'$={author, rank, source :c3, name, affiliation, source :c5, source :c6, source :c0} $\neq$ {source :c0}

{Report}$'$={comments, report, source :c2, target :c1, source :c6, target :c5, source :c0} $\neq$ {source :c0}

{Author, Reviewer}$'$={name, affiliation, source :c5, source :c0} $\neq$ {source :c0}

{Report, Author}$'$={source :c6, source :c0} $\neq$ {source :c0}

{Reviewer, Report}$'$={source :c0} = {source :c0}

{Author, Reviewer, Report}$'$={source :c0} = {source :c0}

Donc, $\sigma(c_{\#7})$=σ ({Reviewer, Author, Report}, {source :c0})= $\frac{2}{8}$=0.25

Ainsi, comme montré via cet exemple, l'utilité de la stabilité peut être observée, d'une part, pour éliminer les nouvelles abstractions découvertes par notre processus et qui sont instables et incorrectes, le cas du concept $c_{\#7}$ et, d'autre part, pour éliminer les abstractions qui existaient dans l'ontologie initiale et qui font partie d'une chaine d'héritage. C'est le cas par exemple des classes *Old_ Person* et *Old_ Woman* qui appartiennent à la chaine d'hériatge illustrée à la section 3.5.2 (voir figure 3.21). En effet, les concepts images de ces classes dans le treillis sont

2. Exemple d'ontologie discuté dans le chapitre précédent.

susceptibles d'avoir des valeurs de stabilité faibles.

Interprétaion dans le contexte ontologique

Dans notre contexte, les objets formels représentent les concepts du domaine dans l'ontologie initiale et un sous-ensemble d'objets formels représente une sous-abstraction de ces concepts du domaines. Par conséquent, la stabilité peut être interprétée comme suit : s'il existe un grand nombre de sous-abstractions possibles des concepts du domaine qui peuvent disparaitre sans perturber le concept concerné (abstraction), ce dernier est insensible aux changements qui peuvent être effectués sur l'ontologie (c'est-à-dire stable), il est donc dit pertinent.

4.3.2 Mesure de densité

Motivation

Le deuxième critère que nous avons choisi pour l'évaluation de la pertinence d'un concept formel consiste à vérifier si le concept est important dans la hiérarchie. En d'autres mots, nous avons besoin de connaitre l'utilité du concept en terme de l'information additionnelle qu'il fournit par rapport aux autres concepts. Pour cela, nous avons utilisé la notion de *densité*.

Description et formulation

L'idée derrière la densité est d'approximer la densité représentationnelle ou le contenu informationnel des classes (Alani et Brewster, 2006).

La mesure peut être appliquée ici pour calculer la densité structurelle d'un concept formel. Nous utilisons la définition de la densité donnée dans (Alani et Brewster, 2006) :

Définition 9. Soient $K=(O,A,I)$ un contexte formel, c_i un concept formel de K et $S=\{S_1, S_2, S_3, S_4, S_5\}=\{attributs(c_i), parents(c_i), enfants(c_i), frères(c_i),$

relations(c_i)}. La densité, *den*, de c_i est définie comme suit :

$$den(c_i) = \sum_{l=1}^{n} w_l \mid S_l \mid \tag{4.2}$$

Où :

$n = \mid S \mid = 5$ et w_l est un facteur de poids (ici fixé à $\frac{1}{5}$, poids égaux).

Dans notre travail, le calcul de la densité est limité aux ensembles suivants :

attributs(c_i) : l'ensemble des attributs de domaine existants dans l'intension de (c_i) qui représentent les propriétés des classes dans l'ontologie initiale.

parents(c_i) : l'ensemble des super-concepts directs de (c_i) dans le treillis.

enfants(c_i) : l'ensemble des sous-concepts directs de (c_i) dans le treillis.

frères(c_i) : l'ensemble des concepts du treillis qui ont les mêmes parents directs que c_i.

relations(c_i) : l'ensemble des relations transversales (liens sémantiques) que le concept c_i possède avec les autres concepts du treillis. Ces relations représentent les *rôles (object properties)* et ils sont déduits, à partir des attributs relationnels de type *« dom :c »* et *« ran :c »*, en visitant itérativement les deux treillis des *Classes* et *Rôles*. Par exemple, s'il existe un rôle r entre deux concepts $c_{\#1}$ et $c_{\#2}$, alors on considère que :

- $c_{\#1}$ possède une relation transversale avec $c_{\#2}$ et
- $c_{\#2}$ possède une relation transversale avec $c_{\#1}$

Le calcul de ces relations transversales est donné par l'algorithme 1.

Interprétation de l'algorithme

L'algorithme 1 prend comme entrée la FTR et retourne en sortie l'ensemble des relations transversales entre les concepts du treillis des *Classes* ainsi que l'ensemble des classes *domaines* et *co-domaines* des concepts du treillis des *Rôles*.

La fonction *ConceptPlusSpecifique(liste de concepts)* permet de déterminer (dans le cas où le nombre d'éléments est supérieur à un) le concept le plus spécifique

de la *liste de concepts*, c'est-à-dire celui qui se trouve au niveau le plus bas de la structure du treillis (celui qui a le nombre minimal d'objets).

Algorithm 1 Générer les relations transversales

Entrées : $FTR = \{T_{Classes}, T_{Roles}\}$

Sorties : les relations transversales entre les concepts de $T_{Classes}$, les classes *domaines* et *co-domaines* des concepts de T_{Roles}

Pour tout concept $c_r \neq bottom \in$ liste des concepts de T_{Roles} **faire**

$ListeDomaines \leftarrow$ liste des concepts c_i référencés dans les attributs relationnels de type « $dom : c_i$ » du concept c_r

$ListeCodomaines \leftarrow$ liste des concepts c_j référencés dans les attributs relationnels de type « $ran : c_j$ » du concept c_r

$c_{domaine}$=*ConceptPlusSpecifique(ListeDomaines)*

$c_{codomaine}$=*ConceptPlusSpecifique(ListeCodomaines)*

Définir une relation transversale entre les deux concepts $c_{domaine}$ et $c_{codomaine}$ dans le treillis des *Classes* :

$relations(c_{domaine}) \leftarrow relations(c_{domaine}) \cup \{c_{codomaine}\}$

$relations(c_{codomaine}) \leftarrow relations(c_{codomaine}) \cup \{c_{domaine}\}$

Définir pour le concept c_r son concept domaine et son concept co-domaine dans le treillis des *Rôles* :

$domain(c_r) \leftarrow c_{domaine}$

$range(c_r) \leftarrow c_{codomaine}$

fin Pour

Exemple :

Reprenons les treillis finaux de l'ontologie CMS à la figure 4.3. Prenons par exemple le concept $c_{\#2}$ du treillis des *Classes*, sa valeur de densité est calculée sur la base des ensembles :

$attributs(c_{\#2})$={contents, paper, title}

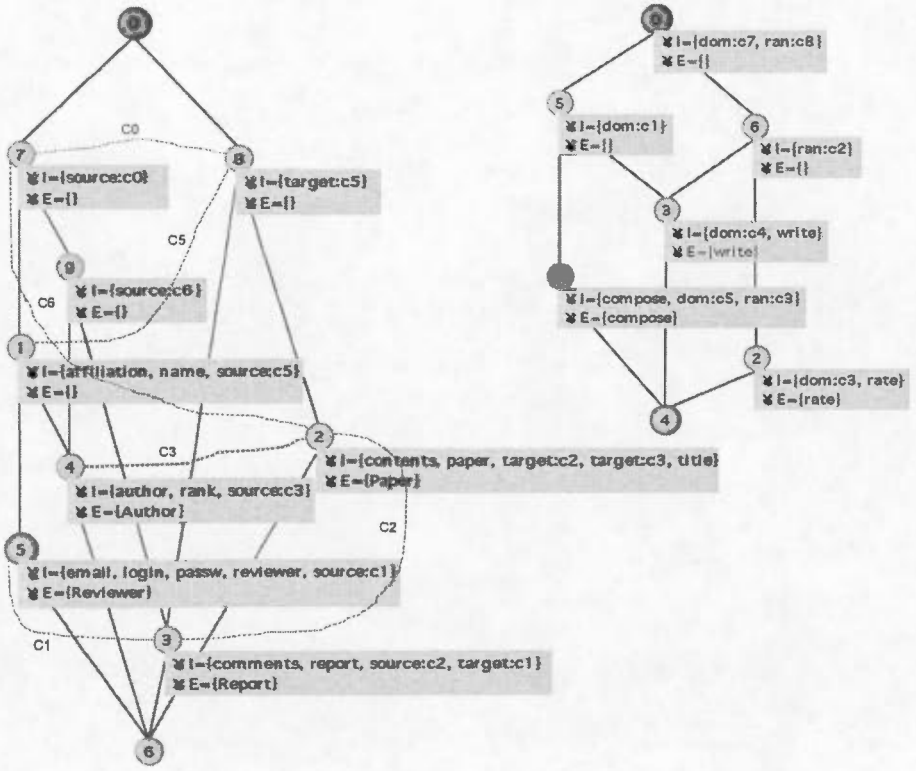


Figure 4.3 L'ensemble des relations transversales (liens pointillés) entre les concepts du treillis des *Classes*.

$parents(c_{\#2})=\{c_{\#8}\}$

$enfants(c_{\#2})=\{c_{\#6}\}$

$frères(c_{\#2})=\{c_{\#3}\}$

$relations(c_{\#2})=\{c_{\#3}, c_{\#4}, c_{\#7}\}$ indiquant que $c_{\#2}$ possède des relations transversales avec les concepts $c_{\#3}$, $c_{\#4}$ et $c_{\#7}$ du treillis.

Donc : $den(c_{\#2})=\frac{1}{5}(3+1+1+1+3) = 1.8$

L'obtention des $relations(c_{\#2})=\{c_{\#3}, c_{\#4}, c_{\#7}\}$ se fait en parcourant le treillis des *Rôles* en appliquant l'algorithme 1. Illustrons, par exemple, l'identification de la relation transversale entre $c_{\#2}$ et $c_{\#4}$:

Pour c_r=c_3 (treillis des *Rôles*) : ListeDomaines=$\{c_{\#4}, c_{\#1}, c_{\#7}\}$ et ListeCodomaines= $\{c_{\#2}, c_{\#8}\}$. En visitant le treillis des classes, $c_{\#4}$, $c_{\#1}$ et $c_{\#7}$ (de même pour $c_{\#2}$ et $c_{\#8}$) sont des concepts comparables, donc :

$C_{domaine}$=ConceptPlusSpecifique($c_{\#4}$, $c_{\#1}$, $c_{\#7}$)=$c_{\#4}$

$C_{codomaine}$=ConceptPlusSpecifique($c_{\#2}$, $c_{\#8}$)=$c_{\#2}$

Ainsi :

Au niveau du treilllis des Classes :

relations($c_{\#4}$)← $relations(c_{\#4}) \cup \{c_{\#2}\}$

relations($c_{\#2}$)← $relations(c_{\#2}) \cup \{c_{\#4}\}$

Au niveau du treilllis des Rôles :

domain($c_{\#3}$)← $c_{\#4}$

range($c_{\#3}$)← $c_{\#2}$

Le résultat d'application de l'algorithme 1 est montré par l'ensemble des relations transversales entres les concepts du treillis des *Classes* à la figure 4.3

Interprétaion dans le contexte ontologique

Dans notre contexte de restructuration, la mesure de densité est bien intéressante, notamment pour évaluer la "richesse" des nouvelles abstractions. En effet, la densité nous informe sur le degré de détail dans la représentation de la connaissance concernant cette abstraction dans l'ontologie restructurée, c'est-à-dire jusqu'à quel point l'abstraction est bien spécifiée, possède une riche description, est suffisamment connectée par des relations sémantiques avec d'autres concepts, etc.

4.3.3 Mesure de proximité sémantique avec les concepts du domaine

Motivation

Comme troisième critère, nous visons à vérifier si un nouveau concept est sus-

ceptible d'être "proche sémantiquement" du centre d'intérêt de l'utilisateur. Nous avons besoin d'évaluer le niveau d'ancrage sémantique, par des liens directs ou indirects, qu'un nouveau concept possède avec les concepts importants dans l'ontologie. Pour ce faire, la notion de *similarité sémantique* appliquée sur un graphe de concepts peut être utilisée.

Description et formulation

La mesure est appliquée ici pour calculer la proximité sémantique entre un concept formel dans le treillis (nouvelle abstraction) et l'ensemble des concepts du domaine (concepts formels représentant les concepts à partir de l'ontologie initiale). Un treillis comme une ontologie peut être vu comme un graphe de concepts et de relations, et donc les mesures de similarité sémantique qui visent à explorer un graphe de concepts pour calculer la distance entre deux éléments peuvent être appliquées. Nous proposons la formule générale suivante :

Définition 10. Soient $K=(O,A,I)$ un contexte formel, c_i un concept formel de K, DC l'ensemble des concepts du domaine et $c_j \in DC$. La proximité sémantique avec les concepts du domaine de c_i est définie comme suit :

$$pscd(c_i) = \frac{1}{m} \sum_{j=1}^{m} proxscd(c_i, c_j) \tag{4.3}$$

Où :

$m =\mid CD \mid$, nombre de concepts du domaine

$proxscd(c_i, c_j)$, proximité sémantique entre un nouveau concept c_i et un concept de domaine c_j

Pour calculer $proxscd(c_i, c_j)$, plusieurs mesures existent dans la littérature et peuvent être appliquées à ce niveau là[3]. Deux alternatives ont été explorées : *mesure de similarité sémantique* et *mesure de degré de relation sémantique.*

3. Une classification de ces mesures est donnée à l'annexe 2

Mesure de similarité sémantique

La façon la plus logique pour mesurer la similarité sémantique entre deux concepts dans une taxonomie ou une ontologie est de calculer la distance qui sépare ces deux concepts. Plusieurs approches se basent sur ce principe (Rada *et al.*, 1989; LEE *et al.*, 1993; Wu et Palmer, 1994; Ehrig *et al.*, 2005; Euzenat et Valtchev, 2004).

Dans notre cas, nous avons choisi d'utiliser la mesure du *plus-court chemin* introduite par (Rada *et al.*, 1989) :

$$simscd(c_i, c_j) = \frac{1}{length(\min \mu : c_i \rightsquigarrow c_j)} \tag{4.4}$$

Où :

$length(\min \mu : c_i \rightsquigarrow c_j)$, la longueur du plus court chemin entre c_i et c_j suivant les relations « is-a ».

Exemple :

Revenons sur l'exemple du treillis des *Classes* à gauche de la figure 4.3. Calculons la similarité sémantique du concept $c_{\#8}$ avec les concepts du domaine $c_{\#2}$, $c_{\#3}$, $c_{\#4}$ et $c_{\#5}$:

$simscd(c_{\#8}, c_{\#2}) = \frac{1}{length(\min \mu : c_{\#8} \rightsquigarrow c_{\#2})} = \frac{1}{1} = 1$

$simscd(c_{\#8}, c_{\#3}) = \frac{1}{length(\min \mu : c_{\#8} \rightsquigarrow c_{\#3})} = \frac{1}{1} = 1$

$simscd(c_{\#8}, c_{\#4}) = \frac{1}{length(\min \mu : c_{\#8} \rightsquigarrow c_{\#4})} = \frac{1}{3} = 0.33$

$simscd(c_{\#8}, c_{\#5}) = \frac{1}{length(\min \mu : c_{\#8} \rightsquigarrow c_{\#5})} = \frac{1}{3} = 0.33$

Donc :

$pscd(c_{\#8}) = \frac{1}{4}(simscd(c_{\#8}, c_{\#2}) + simscd(c_{\#8}, c_{\#3}) + simscd(c_{\#8}, c_{\#4}) + simscd(c_{\#8}, c_{\#5}))$

$pscd(c_{\#8}) = \frac{1}{4}(1 + 1 + 0.33 + 0.33) = 0.66$

Mesure de degré de relation sémantique

Vu que nous cherchons à calculer la proximité sémantique entre deux concepts

formels via non seulement des relations hiérarchiques mais aussi des autres liens sémantiques (ex. les rôles), appelés ici *« relations transversales »*, notre proposition sera guidée par les deux critères suivants :

1. Chercher une mesure qui prend en considération les relations hiérarchiques ainsi que d'autres types de relations sémantiques.

   Selon la définition de (Resnik, 1995), une mesure sémantique peut être définie en utilisant uniquement la relation de spécialisation (on parle alors de mesure de similarité sémantique), ou bien en utilisant tous les types de relations comme la méronymie (part-of) ou tout autre relation (on parle alors de mesure de degré de relation sémantique). Cela permet de considérer les critères fonctionnels décrits par les relations. Par exemple, les concepts de « voiture » et « essence » ont une faible similarité mais un fort degré de relation sémantique (Mazuel et Sabouret, 2007).

   La majorité des travaux actuels dans le domaine se focalisent sur des mesures de similarité qui reposent essentiellement sur l'analyse d'une taxonomie. Ainsi, il existe peu de travaux qui proposent des mesures pouvant utiliser des relations sémantiques non hiérarchiques (Thieu *et al.*, 2004; Yang et Powers, 2005; Mazuel et Sabouret, 2007; Euzenat et Valtchev, 2003).

2. Chercher une mesure qui prend en considération le contenu informationnel en se basant uniquement sur l'ontologie.

   Pour mesurer la similarité sémantique entre deux concepts dans une taxonomie, plusieurs approches se basent sur le principe du calucl de distance qui sépare ces deux concepts (Rada *et al.*, 1989; LEE *et al.*, 1993; Wu et Palmer, 1994; Ehrig *et al.*, 2005). Cependant, ces approches sont problématiques du fait qu'elles supposent que les liens sémantiques dans une taxonomie représentent des distances uniformes (c'est-à-dire, possèdent le même poids). Ce qui rend difficile la définition et le contrôle de la variation

des distances par un seul lien taxonomique (Resnik, 1995).

C'est la raison pour laquelle d'autres travaux ont proposé une autre alternative pour évaluer cette similarité sémantique basée sur la notion du contenu informationnel (Hirst et St-Onge, 1998; Lin, 1998; Resnik, 1999) introduite initialement par (Resnik, 1995). Ainsi, la similarité sémantique entre deux concepts est mesurée par la quantité d'information qu'ils partagent.

Afin de tirer profit des avantages des mesures discutées en 1 et 2, nous nous sommes inspirés de (Mazuel et Sabouret, 2007; Mazuel et Sabouret, 2008) pour appliquer une formule qui peut combiner les deux critères. Les auteurs ont proposé une mesure de degré de relation sémantique dont la formule est :

$$Sim_{ONT}(c_1, c_2) = 1 - \frac{dist_{ONT}(c_1, c_2)}{2 + max_{X \in R} TC_X} \tag{4.5}$$

Où :

R, l'ensemble des relations dans l'ontologie

TC_X, la pondération associée au type d'arrête X

$dist_{ONT}(c_1, c_2)$, la mesure de non-relation sémantique calculée par la formule suivante :

$$dist_{ONT}(c_1, c_2) = \min_{t \in C, X \in R} \{TC_X * (\frac{\mid sp_X(c_1, t) \mid -1}{\mid sp_X(c_1, t) \mid}) + dist_{JCsimple}(t, c_2)\} \tag{4.6}$$

Cette dernière recherche pour chaque relation X quelconque non-hiérarchique le plus court chemin entre c_1 et tous les concepts t ($\mid sp_X(c_1, t) \mid$) atteignables par des liens de type X depuis c_1 et pouvant eux-mêmes atteindre c_2 uniquement par la relation hiérarchique ($dist_{JCsimple}(t, c_2)$).

$dist_{JCsimple}(t, c_2)$, la mesure de similarité initiale de (Jiang et Conrath, 1997) ne prenant en compte que les relations hiérarchiques dont la formule est donnée par :

$$dist_{JCsimple}(t, c_2) = (IC(t) + IC(c_2)) - 2 * IC(ccp(t, c_2)) \tag{4.7}$$

Où :

$ccp(t, c_2)$, plus petit subsumant commun (closest common parent) entre les concepts

t et c_2.

$IC(c)$, une fonction de pondération des nœuds de la hiérarchie. Cette fonction est calculée de façon ascendante comme le propose (Seco *et al.*, 2004). Elle se base uniquement sur la structure de la hiérarchie :

$$IC(c) = 1 - \frac{log(hypo(c) + 1)}{log(max)} \tag{4.8}$$

Où :

$hypo(c) \in N$, le nombre de sous-concepts de c.

max, une constante qui représente le nombre total de concepts qui existent dans la hiérarchie.

La formule $dist_{ONT}(c_1, c_2)$ est intéressante pour deux raisons :

1. Elle permet de prendre en compte les relations sémantiques autres que hiérarchiques.
2. Elle s'appuie sur la mesure de distance de (Jiang et Conrath, 1997) qui est considérée comme la plus efficace pour déterminer la proximité sémantique entre deux concepts. Elle a été évaluée et a présenté de bonnes performances (Budanitsky et Hirst, 2006).

Dans notre contexte, nous considérons un seul autre type de lien sémantique autre que hiérarchique entre les concepts formels, celui des *relations transversales* (RT). Donc, pour calculer le degré de relation sémantique entre un nouveau concept abstrait c_i et un concept de domaine c_j, nous utilisons la formule suivante :

$$drelscd(c_i, c_j) = 1 - \frac{dist_{ONT}(c_i, c_j)}{2 + TC_{RT}} = 1 - \frac{dist_{ONT}(c_i, c_j)}{3} \tag{4.9}$$

Où : $TC_{RT} = 1$

$$dist_{ONT}(c_i, c_j) = \min_{t \in C}\{(\frac{\mid sp_{RT}(c_i, t) \mid -1}{\mid sp_{RT}(c_i, t) \mid}) + dist_{JCsimple}(t, c_j)\} \tag{4.10}$$

$dist_{ONT}(c_i, c_j)$, recherche le plus court chemin entre c_i et tous les concepts formels t ($\mid sp_{RT}(c_i, t) \mid$) atteighables par des relations transversales depuis c_i et pouvant

eux-mêmes atteindre c_j uniquement par la relation hiérarchique.
Dans le cas où les concepts t n'existent pas, la distance entre les deux concepts c_i et c_j (ex. $c_{\#9}$ et $c_{\#4}$) est calculée uniquement par la mesure $dist_{JCsimple}$: $dist_{ONT}(c_i, c_j) = dist_{JCsimple}(c_i, c_j)$.

Exemple :
Calculons pour le concept $c_{\#8}$ son degré de relation sémantique avec par exemple le concept $c_{\#4}$ (voir le treillis des *Classes* à la figure 4.3) :

$drelscd(c_{\#8}, c_{\#4}) = 1 - \frac{dist_{ONT}(c_{\#8}, c_{\#4})}{3}$
$dist_{ONT}(c_{\#8}, c_{\#4}) = \min_{t \in C}\{(\frac{|sp_{RT}(c_{\#8}, t)|-1}{|sp_{RT}(c_{\#8}, t)|}) + dist_{JCsimple}(t, c_{\#4})\}$

Les concepts t pouvant être atteignables par $c_{\#8}$ via les relations transversales et pouvant atteindre $c_{\#4}$ uniquement via des relations hiérarchiques sont : $c_{\#1}$, $c_{\#2}$, $c_{\#3}$, $c_{\#5}$ et $c_{\#7}$.
Montrons le détail de calcul pour $t = c_{\#1}$:
$| sp_{RT}(c_{\#8}, c_{\#1}) | = 1$
$dist_{JCsimple}(c_{\#1}, c_{\#4}) = (IC(c_{\#1}) + IC(c_{\#4})) - 2 * IC(ccp(c_{\#1}, c_{\#4}))$
$IC(c_{\#1}) = 1 - \frac{log(hypo(c_{\#1})+1)}{log(max)} = 1 - \frac{log(3)}{log(10)} = 0.52$
$max = 10, hypo(c_{\#1}) = 2$
$IC(c_{\#4}) = 1 - \frac{log(2)}{log(10)} = 0.69$
$IC(ccp(c_{\#1}, c_{\#4})) = IC(c_{\#7}) = 1 - \frac{log(3)}{log(10)} = 0.52$
$dist_{JCsimple}(c_{\#1}, c_{\#4}) = (0.52 + 0.69) - 2 * 0.52 = 0.17$

Le calcul étant fait pour tout les concepts t, le résultat final donne :

$dist_{ONT}(c_{\#8}, c_{\#4}) = 0.17$ (le minimum)
$drelscd(c_{\#8}, c_{\#4}) = 1 - (\frac{0.17}{3}) = 0.95$

De ce qui précède, les résultats de calcul semblent montrer l'utilité de la mesure *degré de relation sémantique* par rapport à la mesure de *similarité sémantique.*

En effet, pour le concept $c_{\#8}$ qui représente une nouvelle abstraction découverte reconnue comme pertinente, les résultats indiquent une valeur élevée de son degré de relation sémantique (0.95) avec un concept du domaine $c_{\#4}$ alors qu'il possède une similarité sémantique faible (0.33) avec ce même concept. Ceci pourrait bien être en faveur du filtrage du concept $c_{\#8}$ comme pertinent.

Interprétation dans le contexte ontologique

Dans notre contexte, nous pouvons interpréter la poroximité sémantique comme suit : plus les nouvelles abstractions sont proches sémantiquement de l'ensemble des concepts du domaine, l'ontologie restructurée est susceptible de représenter le domaine de manière compacte et cohérente.

4.3.4 Mesure de proximité sémantique entre les concepts enfants

Motivation

Il est admis qu'un modèle ontologique doit respecter le principe de la distance sémantique minimale entre les concepts enfants, c'est-à-dire que les sous-concepts d'un concept donné doivent être proches sémantiquement alors que les concepts non similaires doivent se trouver loin dans la hiérarchie. Ceci a motivé notre choix pour ce quatrième critère dans le but d'approximer la pertinence sémantique d'un concept. Notre intuition est que celui-ci devrait regrouper des sous-concepts similaires. Pour ce faire, la notion de distance ou de similarité sémantique entre éléments dans une hiérarchie de concepts peut être explorée.

Description et formulation

Pour calculer la proximité sémantique entre les concepts enfants, nous proposons la formule générale suivante :

Définition 11. Soient $K=(O,A,I)$ un contexte formel, c_i un concept formel de K. La proximité sémantique entre les concepts enfants (dont le nombre peut être

supérieur à deux), *psce*, de c_i est définie par :

$$psce(c_i) = \frac{1}{n} \sum_{j,k=1}^{p} proxsce(c_j, c_k), j \neq k \tag{4.11}$$

Où :

p, nombre de concepts enfants de c_i

$n = \frac{p*(p-1)}{2}$, nombre de couples possibles de concepts enfants de c_i

$proxsce(c_j, c_k)$, proximité sémantique entre deux concepts enfants c_j et c_k du concept formel c_i.

Reste à trouver la formule de calcul pour $proxsce(c_j, c_k)$. L'idée est de vérifier si les deux sous concepts c_j et c_k sont proches sémantiquement dans la réalité, et par conséquent leur parent commun possède une signification dans le monde réel. Nous proposons de chercher cette proximité sémantique dans une ressource externe.

Les ressources externes que nous pouvons utiliser ici devrait respecter les deux critères suivants :

- Avoir une hiérarchie de classement de concepts complète, claire et sémantiquement correcte.
- Être exploitable, c'est-à-dire possibilité d'interroger la ressource à distance ou bien de récupérer ses données.

Après une documentation et divers tests sur les ressources disponibles (telles que Wordnet[4], Wikipedia[5], DBpedia[6]et Yago[7]), le choix s'est porté sur *DBpedia* et

4. http ://wordnet.princeton.edu/

5. http ://www.wikipedia.org/

6. http ://wiki.dbpedia.org/

7. http ://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

Yago.

DBpedia offre une grande quantité de données classées dans une hiérarchie qui est facilement exploitable.

YAGO propose également une hiérarchie de catégories sous forme d'ontologie, une ontologie claire et précise qui est évaluée à la main.

Wikipedia, bien que très complet et constamment mis à jour, possède une hiérarchie de catégorisation des articles trop chaotique. Il existe trop de catégories sémantiquement peu intéressantes et aucun moyen d'identifier celles intéressantes.

Wordnet s'avère être plus une base lexicale qu'une base de connaissances.

Nous illustrons le principe de notre proposition sur la ressource Yago en deux étapes : *préparation des données* et *pseudo code.*

Préparation des données
YAGO est une base de connaissance sémantique dérivée de Wikipedia, WordNet et Geonames[8]. YAGO possède plus de 10 millions d'entités. Chaque entité est une instance d'une ou plusieurs classes et chaque classe est une sous classe d'une ou plusieurs classes, ce qui créé une hiérarchie. Cette hiérarchie de classes est construite à partir des classes Wikipedia et des Synsets Wordnet. Elle est vérifiée à la main.

Les données Yago (de même pour DBpedia) peuvent être interrogées en ligne via SPARQL (pour SPARQL Protocol and RDF Query Language), via un service

8. http ://www.geonames.org/

web [9], ou bien en téléchargeant les jeux de données [10].

Dans le cadre de notre travail, nous avons opté pour le téléchargement des jeux de données disponibles sur le site sous forme de fichiers rdf et de travailler localement. Pour ce faire, un triplestore rdf [11], qui est une base de données conçue pour le stockage de triplets, est utilisé. Ce type de base de données est optimisé pour le stockage d'un très grand nombre de données et la récupération de ces données se fait à l'aide du langage de requêtes SPARQL [12].

Le triplestore pour lequel nous avons opté est *Fuseki* de la fondation *Apache*. Une fois que les triplets soient téléchargés et stockés dans le serveur *Fuseki*, il est possible de les interroger localement.

L'algorithme 2 montre comment nous proposons de calculer la proximité sémantique entre deux concepts formels c_i et c_j en utilisant une ressource externe qui est Yago :

Interprétation de l'algorithme

Le nombre de niveaux dans la hiérarchie (niveau prédéfini) est paramétrable. Il

---

9. Le service web est un Faceted Web Service qui prend une description XML et génère une réponse XML contenant les données.

10. http ://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/

11. RDF est un modèle de données en triplets (sujet ; prédicat ; objet) qui, pris ensemble, forment un graphe ; et les triplestores rdf sont les bases de données qui permettent de stocker, manipuler et interroger ces graphes.

12. SPARQL est un langage de requête permettant de réaliser des opérations CRUD (pour Create, Retrieve, Update and Delete) sur des données de type RDF. C'est grosso modo l'équivalent du langage SQL pour une base de données relationnelle.

Algorithm 2 Calcul de la proximité sémantique entre deux concepts formels

Entrées : c_i et c_j {deux concepts formels}

Sorties : $proxsce(c_i, c_j)$

$Niv \leftarrow$ niveau prédéfini

$G_{c_i} \leftarrow \emptyset$

$G_{c_j} \leftarrow \emptyset$

$Termes_{c_i} \leftarrow listetermes(c_i)$

$Termes_{c_j} \leftarrow listetermes(c_j)$

$n \leftarrow \mid Termes_{c_i} \mid$

$m \leftarrow \mid Termes_{c_j} \mid$

Pour $k = 1$ à n **faire**

Rechercher le k^{ime} élément de $Termes_{c_i}$ dans l'ontologie *Yago*

$G_k \leftarrow$ les super-catégories de ce terme dans *Yago* jusqu'au niveau prédéfini Niv de la hiérarchie {le résultat sera un graphe rdf}

$G_{c_i} \leftarrow G_{c_i} \cup G_k$ {inclure le graphe G_k dans le graphe final G_{c_i} du premier concept c_i}

fin Pour

Pour $k = 1$ à m **faire**

Rechercher le k^{ime} élément de $Termes_{c_j}$ dans l'ontologie *Yago*

$G_k \leftarrow$ les super-catégories de ce terme dans *Yago* jusqu'au niveau prédéfini Niv de la hiérarchie {le résultat sera un graphe rdf}

$G_{c_j} \leftarrow G_{c_j} \cup G_k$ {inclure le graphe G_k dans le graphe final G_{c_j} du deuxième concept c_j}

fin Pour

$proxsce(c_i, c_j) \leftarrow SimSem(G_{c_i}, G_{c_j})$ {similarité sémantique entre les deux graphes rdf finaux G_{c_i} et G_{c_j}}

```
PREFIX  rdfsn: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX  yago: <http://yago-knowledge.org/resource/>

SELECT DISTINCT ?subclass ?subclass1 ?type
WHERE
  { { SELECT DISTINCT ?type
      WHERE
        { ?type yago:hasPreferredMeaning "author"
        }
    }
OPTIONAL {?type rdfs:subClassOf ?subclass }
OPTIONAL {?subclass rdfs:subClassOf ?subclass1 }
  }
```

Figure 4.4 Exemple d'une requête SPARQL.

serait possible de récupérer la hiérarchie complète mais celle-ci est bien souvent trop importante. Notons que plus le niveau fixé est haut moins les catégories obtenues sont sémantiquement pertinentes (car beaucoup trop abstraites). Il faut donc déterminer un niveau moyen étant pertinent pour notre cas.

Notons que $listetermes(c_i)$ est constituée de la liste des objets de l'extension du concept c_i. Si un nom d'objet est composé de plusieurs termes, il est remplacé par l'ensemble des termes qui le composent.

La récupération de la hiérarchie d'un terme, G_k, se fait à l'aide d'une requête SPARQL. la figure 4.4 montre un exemple simple d'une requête recherchant le terme *author* et récupérant ses super-catégories dans l'ontologie Yago jusqu'à un niveau fixé à 3. Étant donné que, dans notre cas, le niveau est paramétrable, alors la requête sera construite dynamiquement dans le programme Java.

Le résultat de la requête est une structure hiérarchique, G_k, représentée sous forme

d'un graphe rdf qui inclut l'ensemble des super-catégories du k^{ime} terme au niveau de l'ontologie Yago, comme illustré à la figure 4.5. Sur le schéma de cette figure, on peut identifier des hiérarchies de catégories récupérées via des requêtes SPARQL avec un niveau de profondeur fixé à 7 : les classes directement au-dessus du terme recherché sont identifiées par *"type"* (niveau 1), les classes des niveaux 2 à 7 sont des super-catégories récupérées en utilisant des relations de type *"subClassOf"*.

Optional est utilisé dans la requête pour indiquer que la super-classe est récupérée dans le cas où elle existe.

Étant donné qu'un terme peut avoir plusieurs sens, la requête nous retourne le sens par défaut dans l'ontologie identifié par *"hasPreferredMeaning"*.

Il est à préciser que le problème de désambigüisation reste à gérer soit par le choix du sens approprié ou bien par le traitement des sens. Pour Yago, la désambigüisation peut être facilitée du fait qu'elle propose directement des homonymes liés au terme recherché. Le résultat obtenu sera donc plusieurs hiérarchies de classes (une par homonyme).

G_{c_i} est le graphe rdf final représentant le concept c_i. Le graphe final est l'union des différents graphes (pris ensemble) $G_k, k = 1..n$, des n termes respectifs du concept c_i.

Enfin, nous proposons que le calcul de la $SimSem(G_{c_i}, G_{c_j})$ soit basé sur la comparaison des deux graphes G_{c_i} et G_{c_j} suivant deux étapes :

1. Définir l'alignement des deux graphes. Un alignement qui consiste à trouver :
   - tous les nœuds concepts (catégories) qui existent dans les deux graphes G_{c_i} et G_{c_j},
   - toutes les relations reliant ces nœuds concepts communs.
2. Mesurer la similarité entre ces deux graphes G_{c_i} et G_{c_j} en se basant sur le

résultat de l'alignement.

Pour ce faire, des alternatives de solutions existent et peuvent être utilisées ou adaptées, citons par exemple (Montes-y Gómez *et al.*, 2000; Euzenat et Valtchev, 2004). Notre choix s'oriente plus vers (Montes-y Gómez *et al.*, 2000) qui propose une méthode pour la comparaison des éléments de connaissances représentés par des graphes conceptuels [13].

Ce choix est dû à la simple raison qu'un graphe rdf peut être facilement représenté par un graphe conceptuel. De plus, la méthode proposée suit le même principe que notre démarche. En effet, elle consiste à faire en premier un alignement des deux graphes conceptuels G_1 et G_2. Le résultat de l'alignement serait un graphe $G_c = G_1 \cap G_2$ qui représente l'ensemble des sous-graphes maximaux communs entre G_1 et G_2. Pour le calcul de G_c, la méthode proposée dans (Myaeng et López-López, 1992) a été utilisée. Ensuite, une formule simple est utilisée pour mesurer la similarité entre les deux graphes G_1 et G_2 en se basant sur leur graphe intersection G_c.

Ainsi, la description de la mesure *psce* est assez complète. Cependant, un travail complémentaire serait nécessaire pour mettre au point un certain nombre de détails avant l'implémentation complète de la mesure. À date, uniquement la partie concernant la recherche des concepts enfants dans les deux ressources *DBpedia* et *Yago* jusqu'à la récupération de leurs graphes rdf respectifs a été implémentée et expérimentée.

Exemple :

Prenons le concept $c_{\#9}$ du treillis des *Classes* à gauche de la figure 4.3.

$c_{\#9}$=({Author, Report}, {source :c6, source :c0})

13. Un graphe conceptuel est un réseau de noeuds de concepts et de noeuds de relations (Sowa, 1983).

$psce(c_{\#9}) = \frac{1}{n} \sum_{j,k=1}^{p} proxsce(c_j, c_k), j \neq k$

$p = 2, n = 1$

$psce(c_{\#9}) = proxsce(c_{\#4}, c_{\#3})$

Chercher $proxsce(c_{\#4}, c_{\#3})$ dans Yago :

Soit le niveau prédéfini fixé, par exemple, à 7 :

$Termes_{c_{\#4}} = listetermes(c_{\#4}) = \{\text{Author}\}$

$Termes_{c_{\#3}} = listetermes(c_{\#3}) = \{\text{Report}\}$

$l = p = 1$

la figure 4.4 montre un exemple de la requête SPARQL permettant de rechercher le terme *author* dans l'ontologie Yago (avec un niveau égale à 3).

Le résultat de la recherche des deux termes *author* et *report* jusqu'au niveau 7 de profondeur de la hiérarchie est représenté, respectivement, par les deux graphes G_1 et G_2 montrés à la figure 4.5.

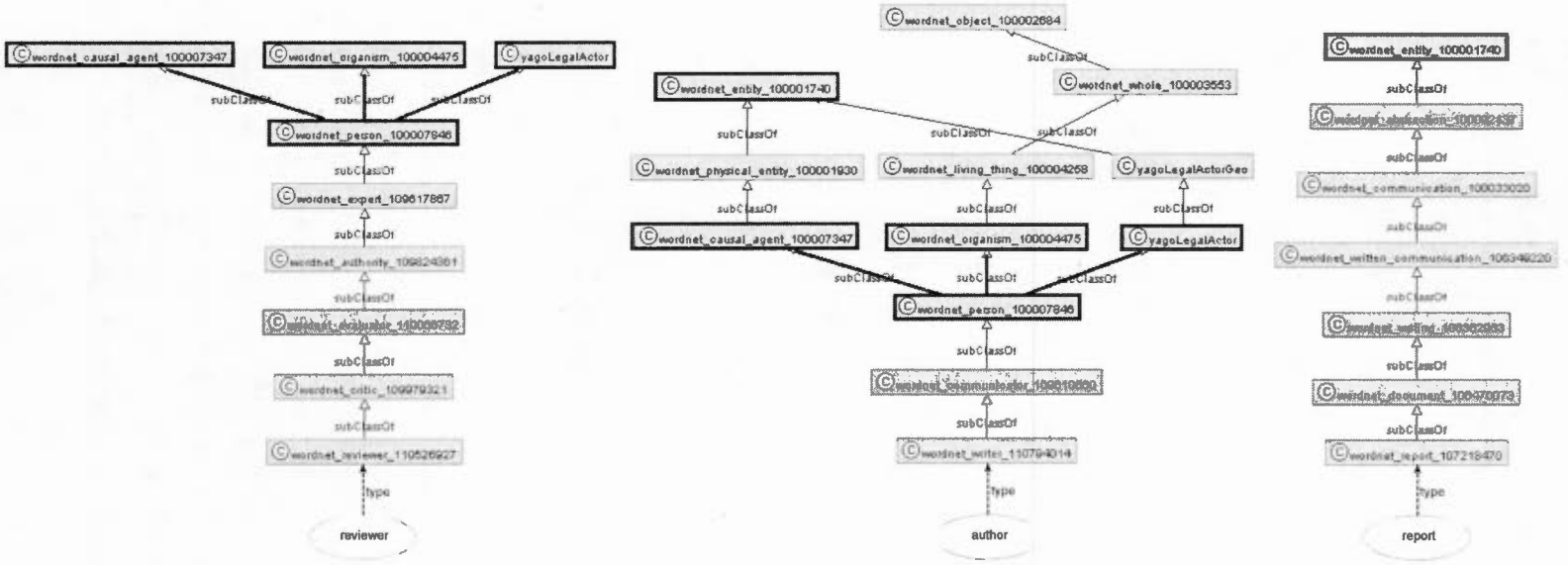


Figure 4.5 Graphes Rdf issus de la recherche des termes respectifs *reviewer*, *author* et *report* dans Yago.

$G_{author} \leftarrow G_1$ (la liste contient un seul graphe rdf du fait que le premier concept possède un seul terme de recherche)
$G_{report} \leftarrow G_2$ (la liste contient un seul graphe rdf du fait que le deuxième concept possède un seul terme de recherche)

Une fois les deux graphes finaux récupérés G_{author} et G_{report} représentant les deux concepts $c_{\#4}$ et $c_{\#3}$, la similarité sémantique entre ces deux graphes rdf sera calculée en appliquant la mesure retenue à cet effet.

En observant les deux graphes, nous pouvons remarquer qu'il n'existe pas vraiment de catégories communes significatives entre eux à l'exception de la catégorie *Entity* qui est une catégorie trop abstraite. Par conséquent, la valeur de la similarité entre ces deux graphes serait très faible indiquant que les deux concepts $c_{\#4}$ et $c_{\#3}$ ne sont pas proches sémantiquement, et donc leur concept parent $c_{\#9}$ n'a pas une sémantique précise. Par contre, les deux concepts représentés par les deux termes *author* et *reviewer* sont évalués comme étant proches sémantiquement car leurs graphes respectifs (illustrés à la figure 4.5) possèdent des catégories et des relations en commun [14].

Ainsi, cet exemple a montré l'application de la mesure de proximité sémantique entre concepts enfants qui peut être utile dans deux cas :

- Supprimer les nouvelles abstractions découvertes non correctes sémantiquement, le cas du concept $c_{\#9}$ de l'ontologie CMS.
- Éliminer les abstractions qui existaient dans l'ontologie initiale et qui sont erronées sémantiquement, le cas par exemple de l'abstraction *Auto* illustrée à la section 3.5.2 (voir figure 3.22) qui regroupe les deux concepts *Vehicle* et *Plant*. Le même algorithme 2 sera appliqué sur les deux concepts *Vehicle*

14. À noter que si on considère un niveau de profondeur plus élevé dans la recherche, le nombre de catégories communes sera augmentée par les super-catégories du niveau supérieur

avec $Termes_{Vehicle} = \{vehicle, car, bus, bicycle\}$ et $Plant$ avec $Termes_{Plant} = \{plant, grass, tree\}$, et la comparaison sémantique sera faite entre leurs deux graphes respectifs $G_{Vehicle} = G_{vehicle} \cup G_{car} \cup G_{bus} \cup G_{bicycle}$ et $G_{Plant} = G_{plant} \cup G_{grass} \cup G_{tree}$. En outre, la démarche adoptée peut facilement être complétée pour corriger davantage d'autres cas d'erreurs sémantiques plus complexes.

Interprétaion dans le contexte ontologique

La projection de cette mesure dans notre contexte de restructuration est simple : plus l'ensemble des concepts enfants sont similaires, plus leur concept parent est bien défini sémantiquement, donc pertinent.

4.4 Algorithme de filtrage

La méthode de filtrage proposée reçoit en entrée la FTR générée par l'approche de restructuration avec l'ARC, exécute une heuristique de filtrage basée sur une combinaison de mesures et retourne en sortie un ensemble de concepts formels (ceux jugés pertinents).

Cette étape consiste à concevoir un algorithme pour le filtrage des treillis. Cet algorithme aura la structure suivante :

Entrées : Famille des treillis relationnels générés par notre approche de restructuration.

Sorties : Famille d'ensembles de concepts filtrés, c'est à dire les concepts formels jugés pertinents qui vont constituer les concepts de l'ontologie restructurée.

Corps : La conception de l'algorithme dépendra des mesures d'évaluation retenues. Il s'agit de trouver la (les) bonne(s) heuristique(s) de filtrage formalisant des règles d'application sur ces mesures.

Notre algorithme de filtrage sera appliqué sur la FTR en deux étapes (voir figure 4.6) :

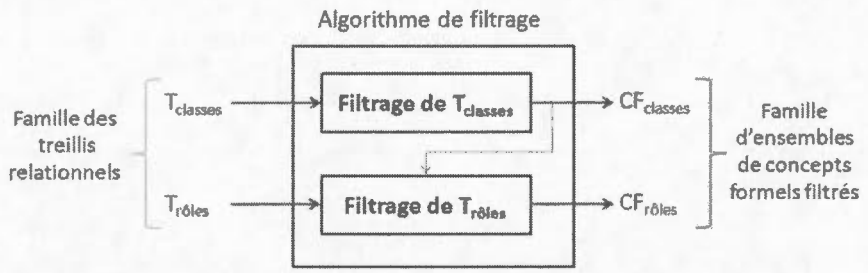


Figure 4.6 Schéma général de l'algorithme de filtrage.

- Filtrage du treillis des *Classes*
- Filtrage du treillis des *Rôles*

Ci-après, sont présentés deux algorithmes 3 et 4 de filtrage des deux treillis de la FTR ($T_{Classes}$ et T_{Roles}), respectivement.

Interprétation des deux algorithmes

Les algorithmes 3 et 4 produisent, à partir des deux treillis $T_{Classes}$ et T_{Roles}, les deux ensembles de concepts formels filtrés respectifs $CF_{Classes}$ et CF_{Roles} en 4 étapes :

1. Identification des concepts non pertinents : Pour le treillis des *Classes*, il s'agit de l'ensemble des concepts formels jugés non pertinents par l'heuristique de filtrage. Pour le treillis des *Rôles*, l'ensemble des concepts non pertinents dépendra de l'ensemble $CF_{Classes}$. En effet, un concept dans le treillis T_{Roles} est considéré comme pertinent s'il implique deux concepts dans $CF_{Classes}$, sinon il est considéré non pertinent.
2. Suppression des concepts jugés non pertinents : Le résultat sera l'ensemble CF_0.
3. Suppression des relations de généralisation non nécessaires : La suppression d'un concept c au niveau du treillis $T_{Classes}$ (ou T_{Roles}) implique la suppression des relations de généralisation dans lesquelles le concept c participe, soit,

Algorithm 3 Filtrage du treillis des *Classes*

Entrées : $T_{Classes}$

Sorties : $CF_{Classes}$

{Identification des concepts non pertinents}

Appliquer l'heuristique de filtrage sur $T_{Classes}$

ConceptsPertinents ← *liste des concepts jugés pertinents par l'heuristique*

ConceptsNonPertinents ← *liste des concepts jugés non pertinents par l'heuristique*

{Suppression des concepts jugés non pertinents}

Pour tout $c \in ConceptsNonPertinents$ **faire**

ListeParents ← *liste des concepts parents de c*

ListeEnfants ← *liste des concepts enfants de c*

Pour tout $p \in ListeParents$ **faire**

$Enfants(p) \leftarrow Enfants(p) \cup ListeEnfants$

fin Pour

Pour tout $e \in ListeEnfants$ **faire**

$Parents(e) \leftarrow Parents(e) \cup ListeParents$

fin Pour

fin Pour

$CF_0 \leftarrow T_{classes} - ConceptsNonPertinents$

{Suppression des relations de généralisation non nécessaires}

$CF_{Classes} \leftarrow CF_0 - RelationsNonNecessaires$

Algorithm 4 Filtrage du treillis des *Rôles*

Entrées : $T_{Classes}, CF_{Classes}$

Sorties : CF_{Roles}

{Identification des concepts non pertinent}

Pour tout $c_r \in T_{Roles}$ **faire**

ListeAttRel $\leftarrow$ attributs relationnels dans l'intension de c_r de type « rel :c »

Pour tout *attrel* $\in$ ListeAttRel **faire**

Récupérer le concept référencé c_i

Si $c_i \notin CF_{Classes}$ **alors**

Supprimer l'attribut relationnel *attrel* de c_r

fin Si

fin Pour

Si c_r nc contient pas (au moins) un attribut de type « dom :c_i » et un autre de type « ran :c_j » **alors**

$ConceptsNonPertinents \leftarrow c_r$

fin Si

fin Pour

{Suppression des concepts jugés non pertinents}

Pour tout $c \in ConceptsNonPertinents$ **faire**

$Parents(c) \leftarrow listedesconceptsparentsdec$

$Enfants(c) \leftarrow listedesconceptsenfantsdec$

Pour tout $p \in Parents(c)$ **faire**

$Enfants(p) \leftarrow Enfants(p) \cup Enfants(c)$

fin Pour

Pour tout $e \in Enfants(c)$ **faire**

$Parents(e) \leftarrow Parents(e) \cup Parents(c)$

fin Pour

fin Pour

$CF_0 \leftarrow T_{roles} - ConceptsNonPertinents$

{Suppression des relations de généralisation non nécessaires}

$CF_{Roles} \leftarrow CF_0 - RelationsNonNecessaires$

IsA(c,c_i) et IsA(c_j,c). Le résultat de cette étape est l'ensemble $CF_{Classes}$ (ou CF_{Roles}).

Les prochaines sous-sections discutent quelques exemples d'heuristiques que nous avons testé empiriquement. Ces tests visaient à trouver la combinaison de mesures qui donnerait la meilleure performance pour notre méthode de filtrage. À ce stade, uniquement les mesures implémentées sont considérées, soit, *stabilité*, *densité* et *proximité sémantique avec les concepts du domaine* avec ses deux alternatives.

Avant d'introduire ces heuristiques de filtrage, notons que les deux mesures de *densité* et de *proximité sémantique* ont déjà été utilisées avec les ontologies (Alani et Brewster, 2006), mais pas la mesure de la *stabilité*. Dans notre travail, nous tentons d'appliquer cette dernière, habituellement utilisée avec les treillis de concepts, dans un contexte ontologique. Pour ce faire, nous avons tout d'abord étudié l'utilité de cette mesure et discuté la corrélation entre la stabilité d'un concept formel et la pertinence de sa traduction en un concept ontologique.

4.4.1 Heuristique basée sur la stabilité

Nous présentons ici une heuristique simple basée uniquement sur la mesure de la *stabilité*. Le principe de cette heuristique est illustré par les règles suivantes : Étant donné un concept formel *cf*,

Règle 1 : Si cardinalité de l'extension(cf) = 1, c'est donc un concept du domaine alors $\sigma(cf)$ est toujours égale à 0.5 (invariable). Le concept est considéré **pertinent**.

Règle 2 : Si cardinalité de l'extension(cf) = 2, alors $\sigma(cf)$ est soit 0.25 (deux sous-concepts) ou 0.5 (un seul sous-concept). Dans ce cas, la valeur de la stabilité n'est pas suffisante pour décider de la pertinence, donc le cas est considéré **non concluant**.

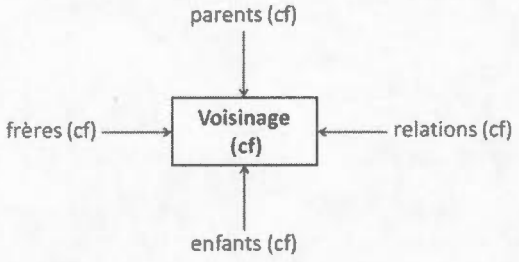


Figure 4.7 L'ensemble des concepts voisins d'un concept formel.

Règle 3 : Si cardinalité de l'extension(cf) $\geqslant 3$, alors $\sigma(cf)$ aura une valeur dans l'intervalle [0,1]. Le critère du seuil est appliqué comme suit :

Rule 3.1 : Si $\sigma(cf) \geqslant 0.5$ le concept (considéré comme stable) est jugé **pertinent**, sinon, il est **non pertinent**.

Notons que le seuil de 0.5 est celui utilisé dans la littérature.

4.4.2 Heuristique basée sur la densité

Le principe de cette heuristique basée uniquement sur la mesure de densité est très simple : Ayant un concept formel *cf*,

Règle 1 : Si $den(cf) \geqslant T1$, alors le concept est jugé **pertinent**, sinon il est **non pertinent**.

Il ne reste que de trouver la bonne valeur pour le seuil $T1$.
La section 5.4.2 montre les résulats des expérimentations mettant sous test cette heuristique dont plusieurs valeurs ont été testées pour le seuil $T1$. La valeur qui a donné les meilleurs résultats vis-à-vis de l'objectif visé par l'heuristique, soit *la médiane des valeurs de densité des concepts voisins*, a été choisie. L'ensemble des concepts voisins d'un concept formel *cf* (voisinage de *cf*) est illustré à la figure 4.7 :

4.4.3 Heuristique basée sur la proximité sémantique avec les concepts du domaine

Cette heuristique suit le même principe que la précédente : Étant donné un concept formel cf,

Règle 1 : Si $pscd(cf) \geqslant T2$, alors le concept est jugé **pertinent**, sinon il est **non pertinent**.

Comme pour l'heuristique précédente, plusieurs valeurs ont été testées pour le seuil $T2$ à la section 5.4.3, et le choix a été fixé sur *la médiane des valeurs de la proximité sémantique de l'ensemble des concepts.*

En outre, pour cette mesure, les deux alternatives de solutions, à savoir la *mesure du plus court chemin* et la *mesure de degré de relation sémantique*, ont été explorées.

4.4.4 Heuristique de filtrage basée sur une combinaison de mesures

Notre objectif ici est d'investiguer un moyen de combiner les trois mesures développées afin d'améliorer les performances de notre méthode de filtrage. Pour ce faire, nous avons commencé par analyser les résultats de l'étude expérimentale effectuée sur les heuristiques basées respectivement sur chacune des mesures à part discutés à la section 5.4.

L'analyse de ces résultats a révélé la situation suivante : En premier lieu, l'heuristique basée sur la *proximité sémantique* est plus avantageuse avec la formule du *degré de relation sémantique* que celle de la *similarité sémantique basée sur le plus court chemin*, et c'est elle qui sera considérée dans la suite de l'analyse. Ensuite, l'heuristique basée sur la *stabilité* a donné de meilleures performances.

Bien que l'hypothèse expérimental d'une bonne corrélation entre *stabilité* et *pertinence* n'est pas valide de façon universelle, si nous restreignons l'évaluation aux concepts formels dont l'extension est constituée d'au moins trois objets formels, la corrélation s'améliore considérablement. En revanche, l'analyse a montré que dans les cas où la *stabilité* n'est pas concluante ou se comporte mal, les deux autres mesures *densité* et *proximité sémantique avec les concepts du domaine* pourraient apporter un plus dans l'évaluation de la pertinence de ces concepts.

Ce constat nous a guidé à examiner une heuristique qui soit basée principalement sur la stabilité en premier lieu, et ensuite chercher à combiner avec elle les deux autres mesures. Le but de cette combinaison est de traiter les cas suivants :

1. *Cas considérés comme non concluants par la stabilité (cardinalité de l'extension = 2),*

2. *Cas des concepts formels pour lesquels la cardinalité de l'extension* $\geqslant 3$ *avec des valeurs faibles de stabilité.*

Nous supposons que pour ces cas, un concept formel est jugé *pertinent* si l'une des conditions suivantes est vérifiée :

- *Cdt1. Le concept est important dans l'hiérarchie : Il apporte plus d'informations par rapport à son voisingae*, ou bien
- *Cdt2. Le concept est proche sémantiquement de l'ensemble des concepts de l'ontologie initiale.*

Ces deux conditions peuvent être interprétées par :

- *den(concept formel)* $\geqslant T1$, ou
- *pscd(concept formel)* $\geqslant T2$.

Sinon, le concept est considéré comme *non pertinent.*

Le principe de cette heuristique est illustré par l'ensemble des règles suivantes : Étant donné un concept formel cf,

Règle 1 : Si cardinalité de l'extension(cf) = 1 -c'est donc un concept du domaine- alors $\sigma(cf)$ est toujours égale à 0.5 (invariable). Le concept est considéré **pertinent**.

Règle 2 : Si cardinalité de l'extension(cf) = 2, alors $\sigma(cf)$ est soit 0.25 (deux sous-concepts) ou 0.5 (un seul sous-concept). Dans ce cas, la valeur de la stabilité n'est pas suffisante pour décider de la pertinence :

Règle 2.1 : Si $(den(cf) \geqslant T1)$ ou $(pscd(cf) \geqslant T2)$, alors le concept est jugé **pertinent**, sinon il est **non pertinent**.

Règle 3 : Si cardinalité de l'extension(cf) $\geqslant 3$, alors $\sigma(cf)$ aura une valeur dans l'intervalle [0,1]. Le critère du seuil est appliqué comme suit :

Rule 3.1 : Si $\sigma(cf) \geqslant 0.5$ le concept (considéré comme stable) est jugé **pertinent**, sinon :

Rule 3.1.1 : Si $(den(cf) \geqslant T1)$ ou $(pscd(cf) \geqslant T2)$, alors le concept est jugé **pertinent**, sinon il est **non pertinent**.

La quatrième mesure, bien qu'elle ne soit pas complètement implémentée, peut être utilisée par la suite dont l'utilité est de vérifier si les concepts gardés par l'heuristique combinant les trois premières mesures sont corrects sémantiquement.

4.5 Analyse de la complexité algorithmique

Cette section présente une brève analyse de la complexité en temps d'exécution de l'algorithme de filtrage proposé.

La complexité asymptotique de notre algorithme, notée $C_{Filtrage}$, est la somme des complexités des deux étapes importantes du filtrage, à savoir *filtrage du treillis des classes* (C_{FTC}) et *filtrage du treillis des rôles* (C_{FTR}) :

$$C_{Filtrage} = C_{FTC} + C_{FTR}$$

La complexité de l'algorithme de filtrage du treillis des classe (Algorithme 3) est dominée par l'étape d'application de l'heuristique de filtrage pour l'identification des concepts non pertinents. Le temps pour supprimer les concepts non pertinents et les relations de généralisation non nécessaires est donné par :

$$C = N * (P + E) = O(N * P + N * E) = maximum(O(N * P), O(N * E))$$

Où : N est le nombre de concepts non pertinents, P est le nombre maximum de concepts parents et E est le nombre maximum de concepts enfants.

La complexité de l'heuristique de filtrage, notée C_{HF}, est plus difficile à établir. Elle dépend des formules des mesures d'évaluation de la pertinence utilisées :

$$C_{HF} = C_{\sigma} + C_{Den} + C_{PSCD}$$

Où : C_{σ}, C_{Den} et C_{PSCD} représentent, respectivement, le temps d'exécution nécessaire pour calculer la *stabilité*, la *densité* et la *proximité sémantique avec les concepts du domaine* pour l'ensemble des concepts du treillis.

Calculons la complexité temporelle pour chaque mesure. Premièrement, pour la *mesure de stabilité* et suivant l'algorithme présenté à la figure 4.1, la première boucle "Pour" est faite en un temps linéaire et la deuxième boucle "Tant que" est faite en un temps égal à $n * log(n)$ en fonction du nombre de concepts formels du treillis (n). Donc :

$$C_{\sigma} = n + n * log(n) = maximum(O(n), O(n * log(n))) = O(n * log(n))$$

La complexité de la *mesure de densité* s'écrit comme suit :

$$C_{Den} = n * C_{Den(c)}$$

Où : $C_{Den(c)}$ est le temps estimé pour calculer la densité d'un concept formel qui se fait en un temps constant. Donc :

$$C_{Den} = O(n)$$

La complexité de la *mesure de proximité sémantique avec les concepts du domaine* s'écrit comme suit :

$$C_{PSCD} = n * C_{PSCD(c)}$$

Où : $C_{PSCD(c)}$ peut être approximé par la complexité temporelle pour le calcul du degré de relation sémantique entre deux concepts qui se fait en un temps linéaire, multiplié par le nombre de concepts du domaine : $C_{PSCD(c)} = m * n$. Donc :

$$C_{PSCD} = n * m * n = O(m * n^2)$$

En résumé, la complexité en temps d'exécution de notre heuristique de filtrage est approximée par :

$$C_{HF} = maximum(O(n * log(n)), O(n), O(m * n^2)) = O(m * n^2)$$

Par conséquent,

$$C_{FTC} = O(m * n^2)$$

En suivant la même analyse, la complexité de l'algorithme de filtrage du treillis des rôles (Algorithme 4), dominée par les deux premières boucles "Pour" imbriquées qui servent à identifier les concepts non pertinents, est estimée à :

$$C_{FTR} = O(n_r * A)$$

Où : n_r est le nombre de concepts formels du treillis des rôles et A est le nombre maximum d'attributs formels.

Pour conclure, la complexité asymptotique de notre algorithme de filtrage est définie par :

$$C_{Filtrage} = maximum(O(m * n^2), O(n_r * A)) = O(m * n^2)$$

4.6 Conclusion

Ce chapitre, qui a été entamé par la formulation de la problématique du filtrage adressé par cette thèse, a présenté les solutions proposées pour la traiter.

Dans ce contexte, plusieurs mesures ont été discutées pour approximer la pertinence des concepts formels pour qu'ils soient acceptés dans l'ontologie restructurée.

Toutes ces mesures n'étant pas parfaites et ne donnant pas toujours les meilleures performances de façon absolue, des heuristiques de filtrage combinant les mesures implémentées ont été explorées.

La mise sous tests de ces heuristiques et, par conséquent, l'évaluation de notre méthode de filtrage au sein du cadre global de restructuration fait l'objet du prochain chapitre.

CHAPITRE V

ÉVALUATION DE L'APPROCHE

Ce chapitre présente le rapport d'une évaluation expérimentale visant, en particulier, la validation de la contribution de notre méthode de filtrage au sein du cadre global de la restructuration.

En premier, les outils et plateformes supportant notre cadre sont présentés. Ensuite, le protocole appliqué pour la validation de notre méthode de filtrage est défini, suivi de l'explication des différents types d'expérimentation menées. Enfin, nous discutons l'analyse des résultats obtenus.

5.1 Mise en contexte

Le développement de notre approche est supporté par la plateforme *INUKHUK* (Rouane-Hacene *et al.*, 2011) qui est une infrastructure orientée services basée sur l'Analyse relationnelle de concepts offrant un ensemble d'outils pour l'Ingénierie ontologique, en occurrence, des services pour la construction, la modularisation, la fusion et la restructuration des ontologies. Le système Inukhuk est couplé avec la plateforme GALICIA [1] pour assurer les fonctions de l'ARC ainsi qu'avec d'autres

1. http ://galicia.sourceforge.net/

plateformes et API (ex. Jena[2], Wordnet[3], Wikipédia[4], Gate[5], Align API[6], SimPak[7])

Nous nous sommes intéressés, dans le cadre de ce travail, aux services de base offerts pour la restructuration. Étant donné que ces services s'appuient sur les mêmes étapes constituant le processus général de restructuration décrit à la section 3.4, nous nous sommes appropriés les modules disponibles *ONTOAligner*, *RCFModeler*, *RCAEngine* et *ONTODesigner* pour l'accomplissement des tâches respectives suivantes : *alignement*, *codage*, *analyse* et *rétro-codage de l'ontologie restructurée* que nous avons adaptés pour y inclure notre outil de filtrage *RLFDistiller* (voir figure 5.1). Un premier travail (Rouane-Hacene *et al.*, 2010) a validé le framework général de la restructuration par l'ARC. Des expérimentations ont été effectuées sur des ontologies de petites/moyennes tailles et ont donné des résultats satisfaisants en termes de réorganisation et d'identification de nouvelles abstractions. Ceci nous a permis d'être rassurés, dans un premier temps, sur la validité de notre cadre et nous a montré les limites de l'algorithme naïf appliqué pour le filtrage des treillis et le besoin d'autres stratégies de filtrage plus rigoureuses et plus performantes permettant d'améliorer les résultats.

Le but de notre travail est d'améliorer la plateforme existante par le développement d'un nouveau module de filtrage, appelé *RLFDistiller*, implémentant les

---

2. http ://jena.sourceforge.net/

3. http ://wordnet.princeton.edu/

4. http ://www.wikipedia.org

5. http ://gate.ac.uk/ie/

6. http ://alignapi.gforge.inria.fr/

7. http ://www.ifi.uzh.ch/ddis/simpack.html

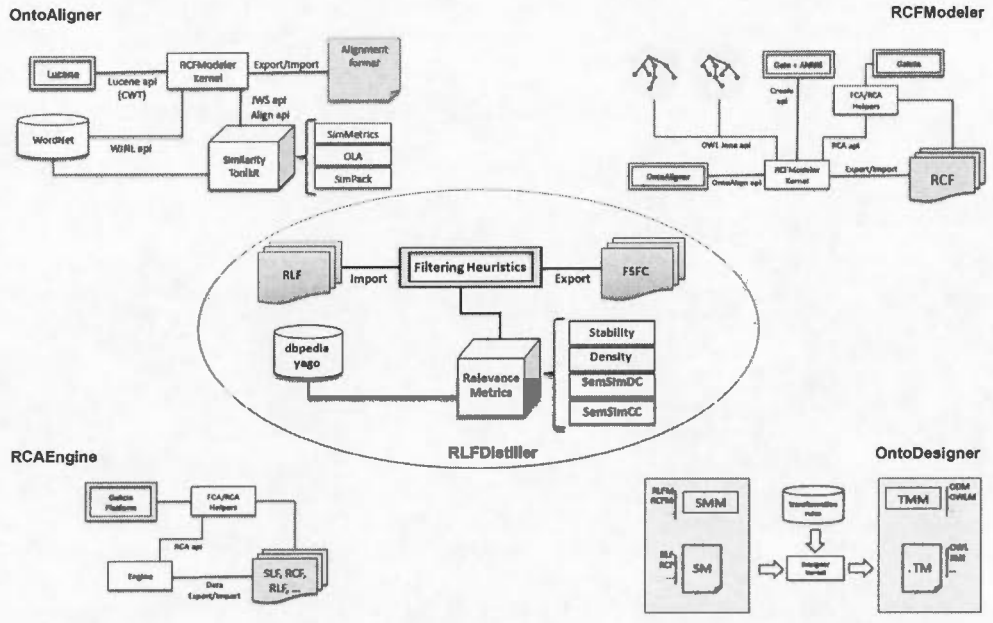


Figure 5.1 Principaux modules de la plateforme *Inukhuk*.

heuristiques basées sur les mesures d'évaluation de la pertinence des concepts formels proposées et discutées dans le chapitre précédent. Trois mesures ont été complètement implémentées : deux pour évaluer la pertinence structurelle, soit, *la stabilité* et *la densité*, et une troisième pour évaluer la pertinence sémantique, soit, *la proximité sémantique avec les concepts du domaine* avec ses deux alternatives (*similarité sémantique* et *degré de relation sémantique*).

Dans ce qui suit, nous présentons notre étude expérimentale visant à tester notre outil de filtrage. Le proptocle de validation suivi est présenté ci-après.

5.2 Protocole de validation

Pour évaluer la pertinence de la restructuration, le protocole optimal serait d'avoir une application basée sur une ontologie et observer le gain de la restructuration

par l'usage de la nouvelle ontologie restructurée. Cependant, nous n'avons pas le cadre qui nous permet de faire ça. C'est la raison pour laquelle, dans notre travail, nous ne comptons pas évaluer la pertinence de la restructuration mais plutôt valider la performance de notre méthode de filtrage au sein du cadre global de la restructuration.

Afin de valider nos heuristiques de filtrage et évaluer l'utilité des mesures développées, deux types de protocoles peuvent être envisagés : (1) *validation centré expert* et (2) *validation (semi-)automatisée.*

5.2.1 Validation centré expert

Une validation appropriée de notre outil devrait suivre le protocole décrit par la démarche suivante :

1. *Choix d'ontologies :* L'idée est de choisir un ensemble d'ontologies de mauvaise qualité, de préférence dont les concepteurs sont disponibles.
2. *Construction des treillis relationnels :* À ce niveau, notre outil de restructuration sera appliqué sur chacune des ontologies choisies et les treillis relationnels respectifs seront générés.
3. *Évaluation des concepts formels par les experts :* Dans cette étape, les experts sont amenés à donner leurs jugements sur la pertinence de chaque concept formel. C'est sur la base de cette pertinence que nous allons décider si le concept formel devrait être élagué ou gardé, et donc ferait partie de l'ontologie restructurée. Nous leurs proposons par exemple de faire une évaluation booléenne : *Pertinent* (à garder) ou *Non pertinent* (à élaguer). Notons que la taille des treillis construits peut constituer une difficulté pour les experts. Une alternative serait de choisir un ensemble représentatif de concepts formels qui seront proposés aux experts pour évaluation.

4. *Application de notre outil de filtrage :* Notre outil de filtrage *RLFDistiller* qui est basée sur nos heuristiques de filtrage sera appliquée sur les treillis relationnels construits pour mesurer la pertinence globale de chaque concept formel. Cet outil retournera en sortie un ensemble de concepts formels filtrés, c'est-à-dire l'ensemble des concepts jugés pertinents par l'outil qui vont constituer les éléments de l'ontologie restructurée. Les autres concepts formels sont considérés comme non pertinents par l'outil.
5. *Corrélation entre les évaluations des experts et la sortie de la technique de filtrage :* Il s'agit de mesurer la corrélation entre les résultats produits par notre technique de filtrage sur l'ensemble des concepts formels avec les jugements des experts sur ce même ensemble.

Cependant, vue la difficulté d'obtenir des ontologies qui sont de mauvaise qualité[8] ainsi que la non-disponibilité des experts de domaines, prêts à collaborer dans ces types d'expérimentations, nous avons opté pour une validation (semi-) automatisée.

5.2.2 Validation (semi-)automatisée

Notre démarche est la suivante :

1. *Choix d'ontologies :* L'idée est de choisir un ensemble d'ontologies de bonne qualité (complètes et sans redondances), qui seront considérées comme des ontologies de référence[9].
2. *Perturbations focalisées des ontologies choisies :* Il s'agit d'introduire volontairement des anomalies pour générer des ontologies de mauvaise qualité.

8. Ces ontologies ne sont pas librement accessibles sur le web.

9. L'analyse de la qualité des ontologies de référence a été faite manuellement pour s'assurer qu'elles ne comportent pas d'erreurs.

Dans notre contexte de validation, nous introduisons des redondances et de l'incomplétude afin de détériorer la qualité des ontologies choisies. L'incomplétude est produite par la suppression d'un ensemble d'abstractions de la hiérarchie de classes, et la propagation des spécifications propres des éléments supprimés vers leurs descendants (sous-classes). Par la même occasion, des redondances de spécifications seront générées dans l'ontologie.

3. *Construction des treillis relationnels :* À ce niveau, l'outil de restructuration sera appliqué sur chacune des ontologies perturbées et les treillis relationnels seront générés. Le treillis relationnel inclura trois types de catégories de concepts formels représentant, respectivement : (1) Classes de l'ontologie initiale (perturbée), (2) Abstractions manquantes redécouvertes par l'outil, et (3) Nouvelles abstractions découvertes par l'outil qui n'ont pas d'image dans l'ontologie de référence.

4. *Application de l'outil de filtrage :* Notre outil de filtrage *RLFDistiller* est appliqué sur les treillis relationnels. Il retourne en sortie un ensemble de concepts formels pertinents qui seront considérés dans l'ontologie restructurée.

5. *Confrontation de chaque ontologie restructurée avec une ontologie de référence :* Pour la confrontation de ces deux ontologies et afin de mesurer *l'exactitude* et la *complétude* de notre approche, nous utilisons les métriques *Précision* et *Rappel* ainsi que *F-mesure* qui combine les deux.

Ces métriques ont été largement utilisées en Recherche d'information (Rijsbergen, 1975). La *précision* représente la probabilité qu'un concept filtré jugé pertinent par l'outil est effectivement pertinent. Elle peut être interprétée comme la probabilité moyenne d'un filtrage exact. Le *rappel* représente la probabilité qu'un concept pertinent qui existe dans l'ontologie de référence soit retenu par l'outil. Il peut être interprété comme la probabilité moyenne d'un filtrage exhaustif. *F-mesure*

est la moyenne harmonique de la *précision* et du *rappel*, où sa meilleure valeur est atteinte à 1 et sa pire valeur à 0.

Notons que les deux métriques *Précision* et *Rappel* nécessitent le calcul des ensembles suivants :

- **Vrai Positifs (VP) :** Concepts dans l'ontologie restructurée qui ont une image ou un équivalent dans l'ontologie de référence.
- **Faux Positifs (FP) :** Concepts dans l'ontologie restructurée qui n'ont pas d'équivalent dans l'ontologie de référence.
- **Vrai Négatifs (VN) :** Concepts formels dans le treillis relationnel jugés non pertinents par l'outil de filtrage qui n'ont pas d'équivalent dans l'ontologie de référence.
- **Faux Négatifs (FN) :** Concepts formels dans le treillis relationnel considérés non pertinents par l'outil de filtrage alors qu'ils possèdent une image dans l'ontologie de référence.

Dans notre contexte, les deux métriques sont définies comme suit :

Définition 12. Ayant une ontologie de référence O_{ref}, la *précision* d'une ontologie restructurée O_{rest} est donnée par la formule suivante :

$$P(O_{rest}, O_{ref}) = \frac{\mid O_{ref} \cap O_{rest} \mid}{\mid O_{rest} \mid} \tag{5.1}$$

Où :

$P(O_{rest}, O_{ref})$: représente le nombre de concepts ontologiques pertinents retrouvés (VP) par rapport au nombre total de concepts proposés par l'outil dans l'ontologie restructurée O_{rest} (VP+FP). En d'autres termes, la *précision* nous permettra de mesurer *l'exactitude* de notre méthode.

Définition 13. Ayant une ontologie de référence O_{ref}, le *rappel* d'une ontologie restructurée O_{rest} est donné par la formule suivante :

$$R(O_{rest}, O_{ref}) = \frac{\mid O_{ref} \cap O_{rest} \mid}{\mid O_{ref} \mid} \tag{5.2}$$

Tableau 5.1 Statistiques sur des *ontologies de référence.*

Ontologie	Classes #	Object prop. #	Data prop. #
CMS	6	5	10
Travel	34	6	7
People	60	14	6
Tourism	76	26	33

Où :

$R(O_{rest}, O_{ref})$: représente le nombre de concepts ontologiques pertinents retrouvés (VP) par rapport au nombre de concepts pertinents existants dans l'ontologie de référence O_{ref} (VP+FN). En d'autres termes, le *rappel* nous permettra de mesurer la *complétude* de notre méthode.

La formule de F-mesure est donnée par :

$$F(O_{rest}, O_{ref}) = 2 * (\frac{(P(O_{rest}, O_{ref}) * R(O_{rest}, O_{ref}))}{(P(O_{rest}, O_{ref}) + R(O_{rest}, O_{ref}))}) \tag{5.3}$$

5.3 Expérimentations

Nous avons appliqué le protocole décrit ci-avant sur un certain nombre d'ontologies de petites/moyennes tailles. Le tableau 5.1 donne des statistiques de base sur ces ontologies (*de référence*).

Le tableau 5.2 donne le nombre de perturbations effectuées sur les ontologies du tableau 5.1 menant à la détérioration de leur qualité en termes de redondances et d'incomplétude. Des exemples de ces perturbations sont illustéres dans le tableau 5.3.

Les objectifs visés par la validation de nos heuristiques sont résumés dans les points suivants :

Tableau 5.2 Nombre de perturbations effectuées sur chaque ontologie.

Ontologie	CMS	Travel	People	Tourism
Nb. perturbations	2	2	3	4

Tableau 5.3 Exemples de perturbations effectuées sur des *ontologies de référence.*

Ontologie	Perturbation
Travel	1. Suppression de la classe *RuralArea* et déplacement de sa propriété *id_RuralArea* aux sous-classes. 2. Suppression de la classe *Activity* et déplacement de ses propriétés *hasActivity* et *isOfferedAt* aux sous-classes.
People	1. Suppression de la classe *Company* et déplacement de sa propriété *id_company* aux sous-classes. 2. Suppression de la classe *Publication* et déplacement de ses propriétés *id_publication* et *reads* aux sous-classes.

- Sélection de tous les concepts du domaine, c'est-à-dire les concepts dans l'ontologie perturbée.
- Sélection des abstractions supprimées dans l'ontologie perturbée et redécouvertes par le cadre de restructuration.
- Élagage des nouvelles abstractions découvertes par le cadre et qui n'ont pas d'équivalent dans l'ontologie de référence.

Quatre types d'expérimentations ont été menées :

Expérimentation 1 :

Cette première expérimentation vise à tester l'heuristique basée sur la mesure de la *stabilité* définie à la section 4.4.1. Le but est d'étudier la corrélation entre la stabilité d'un concept formel et sa pertinence en tant qu'un concept ontologique. Il s'agit de vérifier si :

- tous les concepts du domaine ont des valeurs élevées de stabilité ($\geqslant 0.5$).
- les abstractions redécouvertes ont des valeurs élevées de stabilité ($\geqslant 0.5$).
- les autres nouvelles abstractions découvertes superflues ont des valeurs faibles de stabilité (< 0.5).

Expérimentation 2 :

Dans la même perspective de l'expérimentation 1, nous testons l'heuristique basée sur la mesure de la *densité* définie à la section 4.4.2. En outre, plusieurs tests de la valeur appropriée pour le seuil $T1$ utilisé avec cette mesure sont explorés.

Expérimentation 3 :

Ce troisième type d'expérimentation est consacré à l'évaluation de l'heuristique basée sur la mesure de la *proximité sémantique avec les concepts du domaine* définie à la section 4.4.3. En premier, sont testées certaines valeurs pour le seuil $T2$. Ensuite, les deux alternatives, à savoir (1) *la similarité sémantique basée sur le calcul du plus court chemin*, et (2) *le degré de relation sémantique*, sont explorées avec cette heuristique et les résultats de performance sont discutées en conséquence.

Expérimentation 4 :

Enfin, cette dernière expérimentation vise la validation de l'heuristique basée sur une combinaison de mesures telle que définie à la section 4.4.4. Trois scénarios incrémentaux sont considérés :

1. **Scénario 1.** Aucune règle n'est appliquée et l'algorithme naïf de filtrage des treillis initialement implémenté est utilisé.

2. **Scénario 2.** Uniquement la mesure de stabilité est considérée ce qui signifie que les deux règles 2.1 et 3.1.1 sont éliminées.

3. **Scénario 3.** Les mesures de *densité* et *proximité sémantique* sont combinées avec la mesure de *stabilité* et donc toutes les règles décrites dans l'heuristique

Tableau 5.4 Résultats finaux de l'expérimentation 1.

Ontologie	VP	FP	VN	FN	Précision	Rappel	F-mesure
CMS	4	0	2	2	1	0.66	0.80
Travel	33	0	3	1	1	0.97	0.98
People	59	2	3	1	0.96	0.98	0.97
Tourism	75	9	21	1	0.89	0.98	0.93

sont appliquées.

5.4 Analyse et discussion des résultats

5.4.1 Résultats de l'expérimentation 1 : Évaluation de l'heuristique basée sur la mesure de *stabilité*

Les résultats de notre étude expérimentale sont résumées dans le tableau 5.4 qui donne, pour chaque ontologie, les cardinalités respectives pour les quatre ensembles définis précédemment (VP, FP, VN et FN). Il cite aussi les valeurs des métriques de la qualité (*Précision*, *Rappel* et *F-mesure*).

Dans ce qui suit, nous donnons des exemples pour les quatre ensembles :

- **VP :** Dans le treillis relationnel de l'ontologie *Travel* :
  - Les concepts $c_{\#5}$ et $c_{\#23}$ (voir figure 5.2) qui représentent deux abstractions dans l'ontologie perturbée (Destination et UrbanArea, respectivement) sont jugés pertinents par l'outil avec des valeurs élevées de stabilité (0.91 et 0.68, respectivement).
  - Le concept $c_{\#43}$ (voir figure 5.3) représentant l'abstraction supprimée Activity a été découvert par factorisation de propriétés communes (object properties). Pour sa valeur élevée de stabilité, le concept est considéré comme pertinent

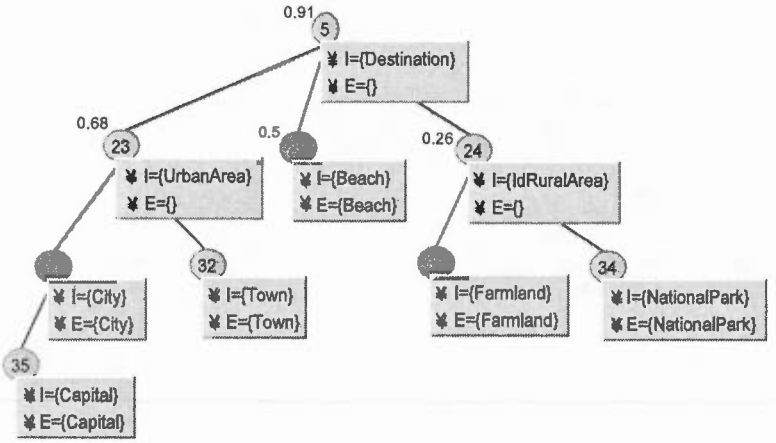


Figure 5.2 Une partie du treillis de concepts construit à partir de la version perturbée de l'ontologie *Travel*.

par l'outil.

- **VN :** Toujours dans l'ontologie *Travel* (voir figure 5.3) les concepts $c_{\#45}$, $c_{\#42}$ et $c_{\#46}$ représentent de nouvelles abstractions découvertes. Dans le treillis, ils sont des super-concepts de $c_{\#43}$ (Activity). Par exemple, $c_{\#45}$ regroupe Destination et Activity, d'où il correspond à une notion trop générale qui ne semble pas pertinente et donc ne devrait pas être retenue. Les trois concepts sont effectivement jugés non pertinents par l'outil pour leurs valeurs faibles de stabilité (0.46, 0.48 et 0.49, respectivement).
- **FP :** Le concept $c_{\#66}$ dans le treillis associé à l'ontologie *People* (voir figure 5.4) possède une valeur de stabilité 0.75 et, par conséquent, il est considéré pertinent par l'outil. Cependant, l'abstraction qu'il représente est très générale et donc elle n'existe pas dans l'ontologie de référence.
- **FN :** Le concept $c_{\#8}$ dans l'ontologie *People* (voir figure 5.4) représente l'abstraction supprimée Publication qui a été redécouverte. Donc, c'est un concept légitime qui devrait être gardé dans l'ontologie restructurée. Cependant, il est

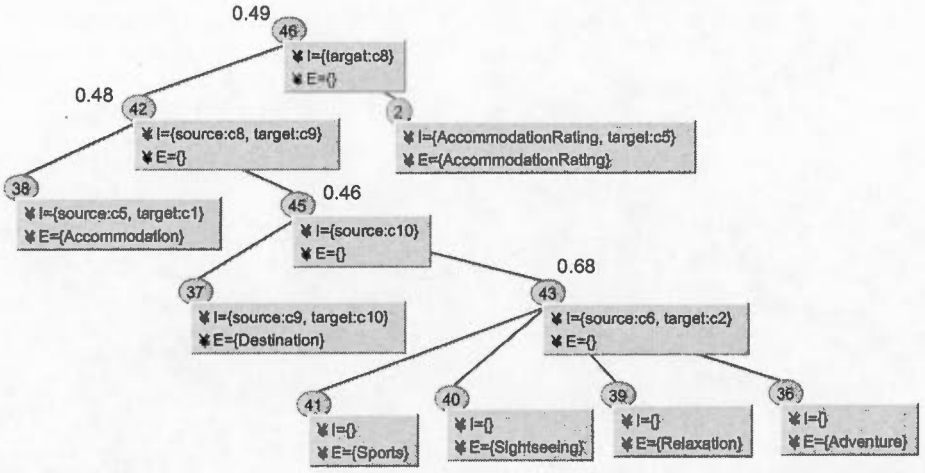


Figure 5.3 Une autre partie du treillis de concepts construit à partir de la version perturbée de l'ontologie *Travel*.

considéré comme non pertinent par l'outil puisque sa valeur de stabilité 0.46 est inférieure au seuil.

Discussion

Après analyse des résultats précédents, les conclusions suivantes peuvent être faites :

En premier, il est clair qu'une parfaite corrélation entre la stabilité d'un concept formel et sa pertinence ne serait pas réaliste. Aussi, la stabilité ne peut pas être utilisée pour ordonner les concepts par leur pertinence : une valeur de stabilité élevée ne signifie pas nécessairement une meilleure pertinence.

Sur une note plus positive, 100% de concepts dans les ontologies perturbées ont eu des valeurs élevées de stabilité et par conséquent ont été gardés par l'outil comme pertinents. Ceci parle en faveur de l'utilisation de cette mesure dans l'heuristique de filtrage visée, possiblement complétée par d'autres mesures.

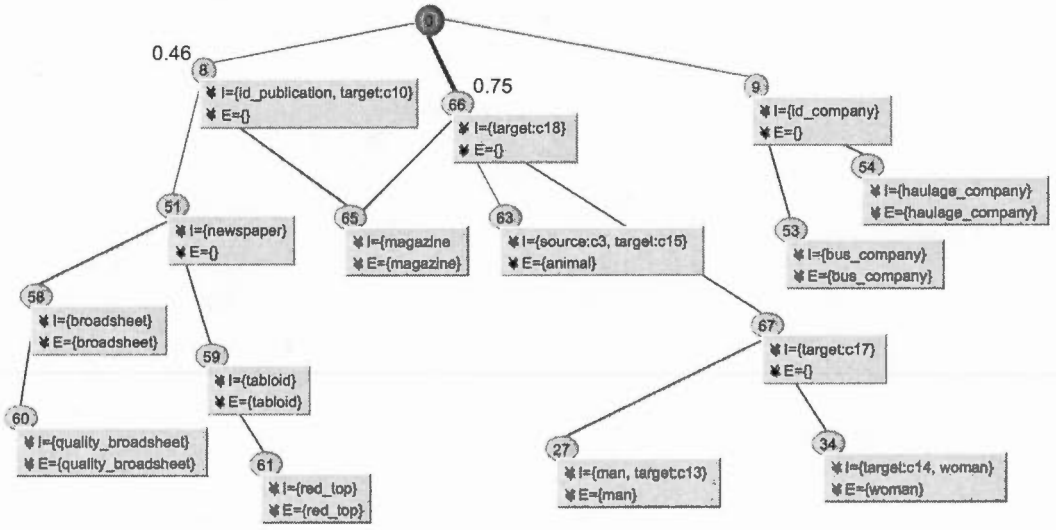


Figure 5.4 Une partie du treillis de concepts construit à partir de la version perturbée de l'ontologie *People*.

Ensuite, 37.5% des abstractions redécouvertes ont été jugées pertinentes. Ce taux peut sembler faible, mais il reste un pourcentage de 50% considérées comme des cas non concluants. En théorie, ces cas pourraient avoir la chance d'être sélectionnés par d'autres mesures dans l'outil de filtrage final. Donc, le taux réel de filtrage des abstractions manquantes, dans le cas de la présente étude, est compris entre 37.5% et 87.5%.

Finalement, les abstractions superflues découvertes par l'ARC ont été élaguées par l'outil avec un taux de 51.25%, alors que 24.39% ont été considérées comme des cas non concluants (encore une fois, ces cas pourraient être élaguées par d'autres mesures). Donc, le taux global d'élagage des abstractions non pertinentes varie entre 51.25% et 75.64%.

Tableau 5.5 Résultats finaux de l'expérimentation 2.

Ontologie	T1=Médiane	T1=Moyenne
CMS	0.54	0.33
Travel	0.60	0.24
People	0.84	0.24
Tourism	0.54	0.25

5.4.2 Résultats de l'expérimentation 2 : Évaluation de l'heuristique basée sur la mesure de *densité*

Cette deuxième expérimentation a donné lieu aux résultats présentés dans le tableau 5.5. Ce tableau résume les valeurs de la métrique *F-mesure* pour chaque ontologie du tableau 5.1, et ce pour certaines valeurs de seuil expérimentées.

Pour l'application de la mesure de *densité*, les valeurs possibles du seuil que nous avons choisies de tester sont : *Médiane* et *Moyenne*.

Pour ces valeurs, nous avons pris le choix qu'elles soient calculées sur l'ensemble des valeurs de densité des concepts voisins du concept concerné. Nous supposons simplement que, dans le cas de la *densité*, le niveau de connaissances représenté par un concept dans une hiérarchie dépendra du niveau de connaissances représenté par ses concepts voisins. Cependant, ceci n'exclut pas la possibilité de les calculer sur l'ensemble des valeurs de densité de tous les concepts du treillis.

Discussion

Le tableau 5.5 révèle bien qu'avec une valeur de seuil égale à la *moyenne*, les valeurs de F-mesure sont faibles en comparaison avec celles données par l'heuristique avec une valeur de seuil égale à la *médiane*. Nous retenons donc cette dernière pour le seuil T1.

Tableau 5.6 Résultats finaux de l'expérimentation 3.

Ontologie	T2=0.5	T2=0.75	T2=Méd.	T2=Moy.
CMS	0.40	-	0.50	0.33
Travel	-	-	0.69	0.50
People	-	-	0.65	0.67
Tourism	-	-	0.61	0.52

5.4.3 Résultats de l'expérimentation 3 : Évaluation de l'heuristique basée sur la mesure de *proximité sémantique avec les concepts du domaine*

Pour ce troisième type d'expérimentation, la première partie suit le même principe que l'expérimentation 2 pour valider une valeur pour le seuil T2 utilisé au sein de l'heuristique discutée à la section 4.4.3. Le tableau 5.6 résume les résultats obtenus.

Étant donné que la proximité sémantique prenne ses valeurs dans l'intervalle [0,1], d'autres valeurs pour le seuil T2 sont considérées : 0.5 et 0.75. Notons qu'ici les valeurs des seuils *Médiane* et *Moyenne* sont calculées sur l'ensemble des valeurs de tous les concepts du treillis.

Discussion

En observant les valeurs de *F-mesure* dans le tableau 5.6, nous pouvons remarquer que :

- Avec une valeur élevée de seuil égale à 0.75, le nombre de concepts *VP* sera diminué et même tend vers 0, ce qui donne des valeurs non définies (-) de *F-mesure*, et le nombre de *FN* augmentera en conséquence. En effet, l'heuristique tend à élaguer tous les concepts du treillis. Le seuil 0.75 ne peut pas être retenu.

Tableau 5.7 Performances (*F-mesure*) des deux alternatives pour la mesure de *proximité sémantique avec les concepts du domaine.*

Ontologie	*Plus court chemin*	*Degré de relation sémantique*
CMS	0.50	0.36
Travel	0.69	0.80
People	0.65	0.88
Tourism	0.61	0.77

- Concernant les valeurs 0.5, *Médiane* et *Moyenne*, les trois colonnes respectives du tableau montrent que c'est la *médiane* qui a donné les meilleures performances et donc sera retenue pour le seuil T2.

La deuxième partie de cette expérimentation compare les résultats d'application des deux alternatives :

- *Mesure de similarité sémantique basée sur le plus court chemin.*
- *Mesure du degré de relation sémantique.*

Le tableau 5.7 montre que les performances [10] sont en faveur de la mesure du *degré de relation sémantique.*

En résumé, de l'analyse des trois types d'expérimentations 1, 2 et 3, nous pouvons conclure que la *stabilité* semble donner des performances plus satisfaisantes que les autres mesures comme illustré dans le tableau 5.8. En effet, la *stabilité* est la seule mesure qui réalise mieux les trois objectifs visés par notre validation (voir section 5.3) contrairement aux deux autres, la *densité* et la *proximité sémantique avec les concepts du domaine*, qui possèdent l'inconvénient de faire élaguer un nombre relativement élevé de concepts de l'ontologie perturbée.

---

10. Rappelons que la valeur du seuil T2 utilisée avec les deux alternatives est la *médiane des valeurs de tous les concepts du treillis.*

Tableau 5.8 Comparaison des performances des heuristiques (*stabilité*, *densité* et *proximité sémantique avec les concepts du domaine*).

Ontologie	*Stabilité*	*Densité*	*P. Sémantique*
CMS	0.80	0.54	0.36
Travel	0.98	0.60	0.80
People	0.97	0.84	0.88
Tourism	0.93	0.54	0.77

Enfin, ces résultats parlent en faveur de l'utilisation de la mesure de *stabilité*, en premier, dans l'heuristique de filtrage finale, complétée ensuite par les deux autres mesures, notamment, pour décider de la pertinence des cas non conclus par la *stabilité*.

5.4.4 Résultats de l'expérimentation 4 : Évaluation de l'heuristique basée sur une combinaison de mesures

Cette dernière section est un rapport d'une évaluation expérimentale de l'heuristique de filtrage combinant les trois mesures. Notons que pour cette expérimentation, de nouvelles perturbations ont été introduites aux ontologies utilisées dans les trois expérimentations précédentets. De plus, d'autres exemples d'ontologies ont été considérées (voir tableau 5.9). Le tableau 5.10 donne le nombre de perturbations effectuées sur ces ontologies.

Le tableau 5.11 résume les résultats de la quatrième expérimentation pour les trois scénarios. Le tableau nous informe sur les valeurs des trois métriques *précision(P)*, *rappel(R)* et *F-mesure(F)* pour chaque ontologie. Le détail des calculs est donné à l'annexe 3.

Tableau 5.9 Expérimentation 4 : Liste des *ontologies de référence* considérées.

Ontologie	Classes #	Object prop. #	Data prop. #
CMS	6	5	10
Travel	34	6	7
Factbook-ont	44	40	172
Univ-cs	53	25	15
People	60	14	6
Soft-onto	66	46	18
Tourism	76	26	33
Ka	96	60	40
Pizza	100	10	13

Tableau 5.10 Expérimentation 4 : Nombre de perturbations effectuées sur chaque ontologie.

Ontologie	CMS	Travel	Factbook-ont	Univ-cs	People	Soft-onto	Tourism	Ka	Pizza
Nb. perturb.	2	4	4	5	5	6	7	8	9

Discussion

Pour l'ensemble des ontologies, nous pouvons observer que :

L'application du premier scénario a confirmé les performances satisfaisantes de l'outil de restructuration en termes de réorganisation et d'identification de nouvelles abstractions. La figure 5.5 montre que les valeurs de la précision sont relativement faibles par rapport aux deux autres scénarios. En effet, l'une des limites de l'algorithme "naïf" est le nombre élevé de concepts *faux positifs*. Le problème se pose dans la séparation entre les nouvelles abstractions pertinentes et celles superflues.

Tableau 5.11 Résultats de l'expérimentation 4 pour les trois scénarios.

	Scénario 1			Scénario 2			Scénario 3		
Ontologie	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
CMS	0.71	0.83	0.76	1	0.66	0.80	0.75	1	0.85
Travel	0.81	0.88	0.84	0.84	0.94	0.88	1	1	1
Factbook-ont	0.74	0.90	0.81	0.86	0.90	0.88	0.89	1	0.94
Univ-cs	0.81	0.90	0.85	0.81	0.90	0.85	0.88	0.98	0.92
People	0.66	0.91	0.76	0.84	0.93	0.88	0.98	0.96	0.97
Soft-onto	0.70	0.90	0.79	0.83	0.90	0.86	0.88	0.95	0.91
Tourism	0.62	0.92	0.74	0.71	0.92	0.80	0.93	0.96	0.94
Ka	0.79	0.91	0.85	0.79	0.93	0.86	0.87	0.96	0.92
Pizza	0.68	0.91	0.78	0.74	0.94	0.83	0.81	0.96	0.88

Les résultats d'application du deuxième scénario peuvent être expliqués comme suit : l'utilisation de la mesure de stabilité a permis de sélectionner davantage en moyenne 14.58% des concepts *faux négatifs* et d'élaguer 34.02% des concepts *faux positifs*, ce qui est bien représenté par les valeurs améliorées des deux métriques *précision* et *rappel* par rapport au premier scénario (voir les deux figures 5.5 et 5.6). Notons que le reste, à savoir 85,42% des *faux négatifs* (non filtrés) ainsi que 65.98% des *faux positifs* (non élagués) incluent les cas non concluants.

Concernant les résultats d'application du troisième scénario, nous pouvons constater que les valeurs de la *précision* et du *rappel* ont été améliorées en comparaison avec les deux autres scénarios (voir figures 5.5 et 5.6). Ces résultats sont justifiés, en particulier, par le traitement des cas non concluants pour la stabilité, par les deux autres mesures :

- Les valeurs de la *précision* sont comprises entre 0.75 et 1. Ceci signifie que plus

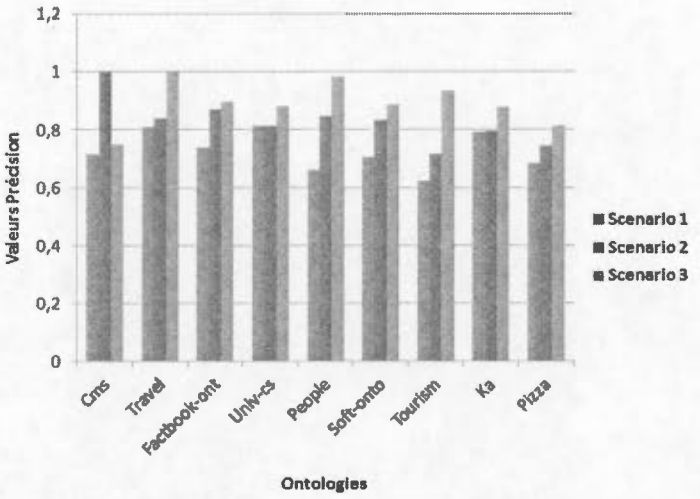


Figure 5.5 Comparaison des valeurs de la métrique *Précision* résultantes de l'application des trois scénarios.

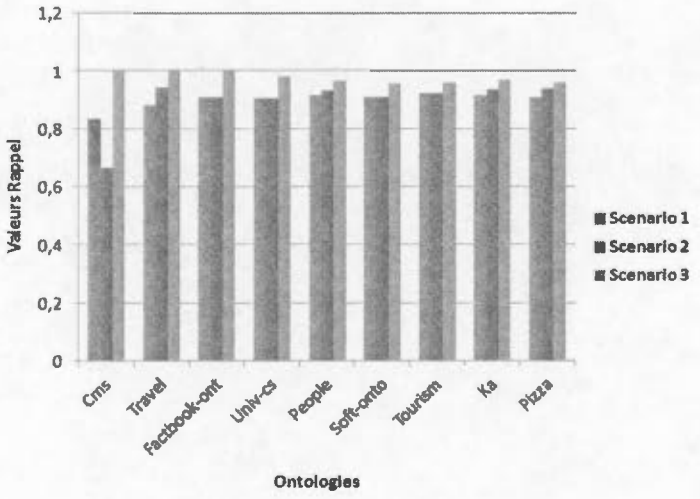


Figure 5.6 Comparaison des valeurs de la métrique *Rappel* résultantes de l'application des trois scénarios.

de 75% des concepts filtrés (jugés pertinents par l'heuristique) sont réellement pertinents ; et donc le pourcentage du bruit [11] introduit par le filtrage dans les ontologies restructurées est compris entre 0% et 25%.

- Les valeurs du *rappel* sont entre 0.95 et 1, ce qui signifie que pratiquement tous les concepts pertinents dans l'ontologie de référence ont été considérés comme pertinents par l'outil. Notons que malgré le bruit introduit dans les ontologies restructurées (les abstractions superflues retenues), la complétude de l'outil parle en faveur de notre méthode en terme du filtrage, d'une part, des concepts de l'ontologie initiale et, d'autre part, des nouvelles abstractions pertinentes.
- Pour chaque ontologie, une valeur élevée de *F-mesure* entre 0.85 et 1 (un bon compromis entre *Précision* et *Rappel*) peut être interprétée comme une bonne restructuration des classes de l'ontologie perturbée en comparaison avec son ontologie de référence. En outre, les valeurs de *F-mesure* montrent que le scénario 3 améliore les performances du processus global de la restructuration par rapport aux scénarios 1 et 2 comme illustré à la figure 5.7.
- Enfin, le troisième scénario semble vérifier les trois objectifs visés par notre validation (voir section 5.3). En effet, notre méthode de filtrage basée sur l'heuristique combinant nos trois mesures permet de :
  - filtrer 100% des concepts de l'ontologie initiale,
  - filtrer en moyenne 68% des nouvelles abstractions pertinentes,
  - élaguer en moyenne 58% des nouvelles abstractions non pertinentes.

---

11. Le bruit ici signifie le nombre de concepts *faux positifs*.

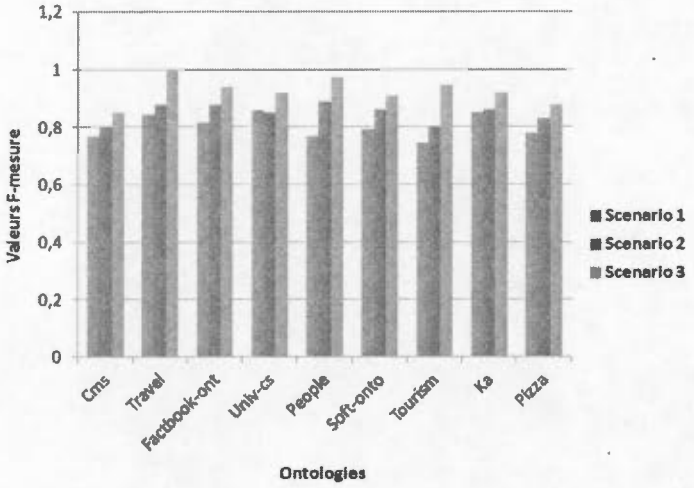


Figure 5.7 Comparaison des valeurs de la métrique *F-mesure* résultantes de l'application des trois scénarios.

5.5 Conclusion

Dans ce chapitre, nous avons évalué la méthode de filtrage proposée dans le chapitre précédent. Nous avons testé les trois mesures retenues au sein d'un ensemble d'heuristiques. Une étude expérimentale a été réalisée sur ces heuristiques impliquant plusieurs ontologies disponibles sur le web.

À partir des résultats discutés, nous pouvons conclure que le filtrage basé sur la combinaison des trois mesures a apporté des améliorations significatives à l'ensemble du processus de restructuration par rapport à l'algorithme naïf ainsi que le filtrage basé, respectivement, sur chacune des mesures.

Les analyses révèlent que tous les concepts dans l'ontologie initiale ont été considérés comme pertinents par l'outil, ce qui est un point positif. En outre, bien que pas encore parfait, notre outil de filtrage se comporte bien dans la séparation entre les nouvelles abstractions pertinentes et celles superflues.

Le prochain et dernier chapitre donne un résumé des réalisations et présente les principales contributions de la thèse, suivis des limites du travail réalisé et des perspectives envisagées dans ce cadre de recherche.

CHAPITRE VI

CONCLUSION

Nous avons présenté dans cette thèse une nouvelle approche de restructuration des ontologies par l'Analyse relationnel de concepts (ARC). Nous nous sommes focalisés sur une étape cruciale du processus de restructuration, à savoir le filtrage des concepts au sein des treillis issus de l'analyse, et nous avons traité le problème d'évaluation de leur pertinence. Étant donné que la pertinence est contextuelle et subjective, nous avons étudié différentes mesures pour l'approximer.

6.1 Contributions

Les principales réalisations et contributions de notre recherche sont résumées comme suit :

Une approche globale pour la restructuration d'ontologies
Comme une contribution à la restructuration des modèles ontologiques, une nouvelle approche basée sur l'ARC est proposée. Une approche qui s'attaque à des problèmes ouverts, rarement abordés dans la littérature du domaine. L'approche proposée se résume comme suit : elle code une ontologie OWL en un ensemble de tables binaires et par conséquent des abstractions de classes et de propriétés, potentiellement pertinentes, pourraient être détectées par l'analyse de concepts. Le résultat de l'analyse est ensuite utilisé pour générer le modèle ontologique

restructuré ; ce nouveau modèle offrira une vue plus complète du domaine.

Dans ce cadre, les contributions spécifiques sont les suivantes :

- L'approche couvre tout le processus de restructuration permettant la prise en compte conjointe des deux aspects suivants. Tout d'abord, elle permet l'identification et la correction des anomalies existantes dans l'ontologie. Ensuite, la découverte et l'intégration des connaissances manquantes. Elle comble ainsi un défaut des approches actuelles qui tentent de résoudre localement des erreurs d'inconsistance et de redondance, et ignorent complètement le problème d'incomplétude, un des objectifs principaux de la restructuration.
- L'approche est complètement automatique. Elle se base sur un cadre formel avec des méthodes universellement applicables qui effectue une réorganisation complète des éléments ontologiques de sorte que les anomalies visées soient détectées et corrigées automatiquement et de manière exhaustive. En comparaison, dans la majorité des travaux, la détection des erreurs est automatique alors que la correction est le plus souvent faite manuellement.
- L'approche fournit, comme pour les classes, une structure hiérarchique des *« propriétés objet »*.
- La méthode de codage proposée, basée sur un méta-modèle simplifié d'ontologies, suit un processus général et extensible. Ce processus de codage permettra donc de représenter et de coder les principaux éléments du méta-modèle OWL complet en une FCR (structure d'entrée de l'ARC).
- L'approche, bien qu'elle soit proposée pour la restructuration des ontologies OWL, peut être appliquée pour d'autres types de langages ontologiques (ex. DAML+OIL). Une adaptation sera nécessaire uniquement au niveau : (1) des règles de codage de l'ontologie représentée dans le langage considéré ; et (2) des règles de traduction des concepts formels filtrés vers le langage ontologique cible.

- Contrairement à la majorité des travaux existants, notre approche ne dépend pas d'un contexte ou d'une ontologie de domaine spécifique, elle est donc généralisable.
- D'un point de vue plus général, notre approche offre un cadre de formalisation pour la génération d'une ontologie OWL à partir d'une représentation conceptuelle (concepts, propriétés, relations, etc.) d'un domaine. Ceci est possible avec une adaptation de la méthode du codage.
- L'étape suivante de notre démarche avait pour but de valider notre framework. Un framework qui intègre des modules implémentant les différentes phases du processus global de la restructuration. Notons qu'à ce stade, le module de filtrage implémentait un algorithme naïf pour le filtrage des treillis et aucune mesure n'était intégrée.

  Des études de cas pratiques ont examiné le potentiel de l'approche (framework global) pour la restructuration d'un certain nombre d'ontologies existantes sur le web.

  Les résultats des expérimentations ont montré des performances satisfaisantes de notre outil en termes de réorganisation des éléments ontologiques ainsi que l'identification de nouvelles abstractions. Les résultats ont également montré les limites de l'algorithme de filtrage de base implémenté et souligné la nécessité d'une nouvelle méthode de filtrage plus efficace.

Une méthode de filtrage des treillis de concepts

Dans notre travail, nous avons aussi proposé une nouvelle stratégie pour le filtrage des concepts pertinents au sein des treillis résultants de l'analyse. Les concepts filtrés constitueront les éléments du modèle ontologique restructuré.

Nous avons choisi de nous focaliser sur cette phase compte tenu de son importance et son influence sur la performance de tout le processus de restructuration. En effet, *l'Analyse relationnelle de concepts* génère des structures de treillis très

complexes qui devraient être nettoyées par l'élagage des concepts non pertinents avant de faire le *rétro-codage* de ces treillis en un modèle ontologique restructuré.

Tout d'abord, nous avons formulé la problématique du filtrage, ensuite construit un moyen pour la résoudre. Notre démarche se résume comme suit : (1) déterminer les critères auxquels doit répondre un concept formel pour qu'il soit accepté comme un concept ontologique pertinent ; (2) traduire chaque critère en une mesure permettant d'évaluer la pertinence relative pour chaque concept formel ; et (3) proposer des heuristiques pour le filtrage des treillis, basées sur les mesures retenues.

Dans ce deuxième cadre de notre recherche, les contributions spécifiques sont :

- La méthode développée représente une solution à la problématique du filtrage des treillis de concepts volumineux dans un contexte de restructuration.
- Des mesures pour l'évaluation de la pertinence relative (structurelle et sémantique) de concepts formels dans un contexte ontologique. Trois mesures ont été implémentées : *stabilité* et *densité* qui contribuent à l'évaluation de la pertinence structurelle d'un concept formel, et la *proximité sémantique avec les concepts du domaine* pour l'évaluation de la pertinence sémantique.
  Tout d'abord, nous avons examiné ces trois mesures et tenté une évaluation de leur utilité vis-à-vis de notre objectif. Dans ce but et pour chaque mesure prise à part, nous avons réalisé un certain nombre d'expérimentations dans lesquelles nous avons appliqué nos outils de restructuration sur des ontologies de mauvaise qualité, appliqué le filtrage basé sur la mesure en question, et enfin comparé les résultats avec une ontologie de référence du même domaine.
  Les résultats des expérimentations ont révélé l'image suivante : l'heuristique basée sur la *stabilité* a donné de meilleures performances. Bien que l'hypothèse expérimentale d'une bonne corrélation entre *stabilité* et *pertinence* n'est pas

valide de façon universelle. Cependant, si nous restreignons l'évaluation aux concepts formels dont l'extent est constitué d'au moins trois objets formels, la corrélation s'améliore considérablement. En revanche, l'analyse des résultats a montré que dans les cas où la *stabilité* n'est pas concluante ou se comporte mal, les deux autres mesures (*densité* et *proximité sémantique*) peuvent apporter un plus dans l'évaluation de la pertinence de ces concepts.

- Une combinaison de mesures qui améliore les performances du processus. Dans la suite de notre étude, nous nous sommes intéressés à la proposition d'heuristiques intégrant les trois mesures implémentées : *stabilité*, *densité* et *proximité sémantique avec les concepts du domaine*, et nous avons tenté de combiner leur application. Une investigation de la meilleure forme pour cette combinaison a été effectuée. Nous avons testé la validité de ces heuristiques en réalisant une étude expérimentale impliquant plusieurs ontologies. Dans les expérimentations menées, nous avons sélectionnés des ontologies disponibles sur le web que nous avons perturbées intentionnellement pour obtenir des ontologies de mauvaise qualité, exécuté nos outils de restructuration, appliqué le filtrage basé sur chaque heuristique, et finalement comparé l'ensemble des concepts résultants considérés comme pertinents avec ceux de l'ontologie de référence.

  D'une part, l'analyse a pu confirmer la bonne forme de la combinaison des mesures (heuristique) qui a donné les performances escomptées vis-à-vis de nos objectifs. En effet, 100% des concepts de l'ontologie initiale sont retenues par l'heuristique. De plus, bien que pas encore parfait, notre outil de filtrage a bien fait la séparation entre les nouvelles abstractions pertinentes et celles qui sont superflues.

  D'autre part, les résultats de l'application de cette heuristique ont permis de conclure qu'un filtrage basé sur une combinaison de mesures apporte une amélioration significative au processus global de la restructuration.

- Les précédents constituent des moyens pour gérer la complexité des treillis. Les

méthodes développées dans cette thèse pourraient être réutilisées dans d'autres contextes d'utilisation de l'AFC et de l'ARC, là où il serait nécessaire de filtrer des treillis de concepts et notamment d'évaluer la pertinence relative des concepts formels au sein de ces structures, citons les autres opérations de l'ingénierie ontologique (construction/apprentissage, fusion, modularisation, etc.) ainsi que le domaine de la réingénierie de modèles.

6.2 Limites et perspectives

Les travaux discutés dans cette thèse comportent encore des éléments pouvant être améliorés et ouvrent plusieurs pistes de recherches. Ces pistes incluent les propositions suivantes qui peuvent être envisagées pour améliorer davantage certaines phases du processus global, à savoir *le codage*, *le filtrage*, *le rétro-codage* ainsi que *le protocole de validation.*

Codage

Dans notre travail, le codage d'une ontologie OWL était guidé par un méta-modèle simplifié. Les entités considérées sont : Concept, ObjectProperty, Datatype, et DatatypeProperty ainsi que les liens hiérarchiques entre les entités et les liens inter-entités (ex. owns, range, domain, etc.). Il sera intéressant d'étendre la méthode du codage pour inclure les autres éléments du méta-modèle complet comme proposé à l'annexe 1.

Une étude détaillée et appropriée de cet aspect doit être effectuée pour établir le sous langage exact OWL qu'on peut utiliser pour bénéficier de l'ARC dans un but d'extraire de nouveaux liens et/ou caractéristiques.

Filtrage

Comme suite au doctorat, il sera complémentaire d'implémenter la mesure qui tente d'aller chercher la *proximité sémantique entre les concepts enfants* d'un

concept formel dans des ressources externes (ex. Dbpedia, Yago). Cette mesure a l'avantage d'apporter un plus dans l'évaluation de la pertinence sémantique des concepts formels, mais peut présenter l'inconvénient de la scalabilité lors de l'exécution. Il sera nécessaire d'étudier la bonne forme d'utilisation de cette mesure et de sa combinaison avec les autres mesures implémentées au sein d'une même heuristique.

Un autre aspect qui n'a pas été traité dans notre processus de filtrage est l'éventuelle corrélation entre les performances des mesures/heuristiques et les caractéristiques des ontologies. En tenir compte permettrait de mieux cibler la fonction du filtrage.

Rétro-codage

Une amélioration peut être apportée à l'étape du *rétro-codage* concernant le nommage des nouvelles abstractions pertinentes. Donner des noms significatifs à ces abstractions sera d'une grande utilité ; tout d'abord, dans l'évaluation de la qualité de l'ontologie restructurée, et ensuite dans sa compréhension et son utilisation.

Protocole de validation

Pour l'évaluation de notre approche, nous nous sommes basés sur un protocole de validation (semi-)automatisé vu, d'une part, la difficulté d'obtenir des ontologies de mauvaise qualité et, d'autre part, le manque d'experts de domaines prêts à collaborer dans les expérimentations. Il sera intéressant de faire une validation centré-expert de notre méthode de filtrage selon le gabarit défini à la section 5.2. Il sera également complémentaire de faire une validation comparée avec les autres approches.

Pour conclure, nous espérons que cette contribution gagnerait à susciter l'intérêt chez d'autres chercheurs pour des développements futurs dans le domaine de l'ingénierie ontologique et de la restructuration de modèles utilisant comme cadre

théorique l'AFC ou l'ARC.

ANNEXE 1

Codage d'ontologies OWL en FCR

Nous proposons un codage guidé par un méta-modèle d'ontologie ODM complet. L'identification des éléments ontologiques à coder se base principalement sur les deux diagrammes suivants : (1) *diagramme des descriptions de classes OWL* (voir figure .1), et (2) *diagramme des descriptions de propriétés OWL* (voir figure .2).

Les entités qui peuvent être considérées sont :

Diagramme des classes OWL :

- *Classes* : OWLClass
- *Restrictions* : OWLRestriction
- *Constructeurs de classes* : ComplementClass, IntersectionClass, UnionClass
- *Liens entre OWLClass* : EquivalentClass, DisjointClass, IntersectionClassForIntersection, UnionClassForUnion et ComplementClassForComplement

Diagramme des proprieties OWL :

- *Propriétés* : Property, OWLDatatypeProperty, OWLObjectProperty, FunctionalProperty
- *Liens entre propriétés* : EquivalentProperty
- *Liens entre ObjectProperties* : InverseProperty
- *Caractéristiques de ObjectProperties* : InverseFunctionalProperty, SymmetricProperty, TransitiveProperty

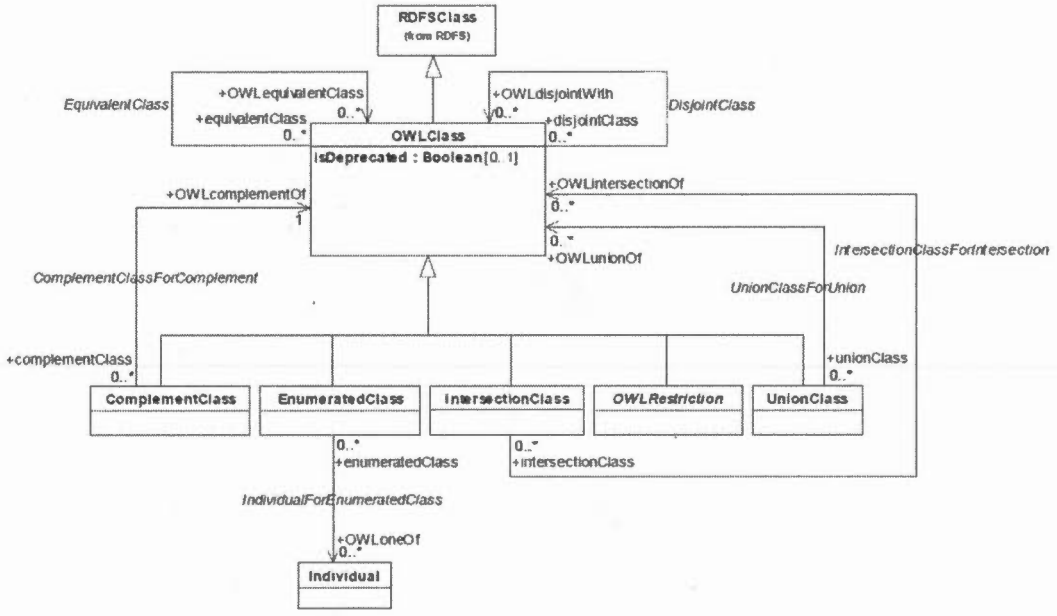


Figure .1 Diagramme des descriptions de classes OWL.

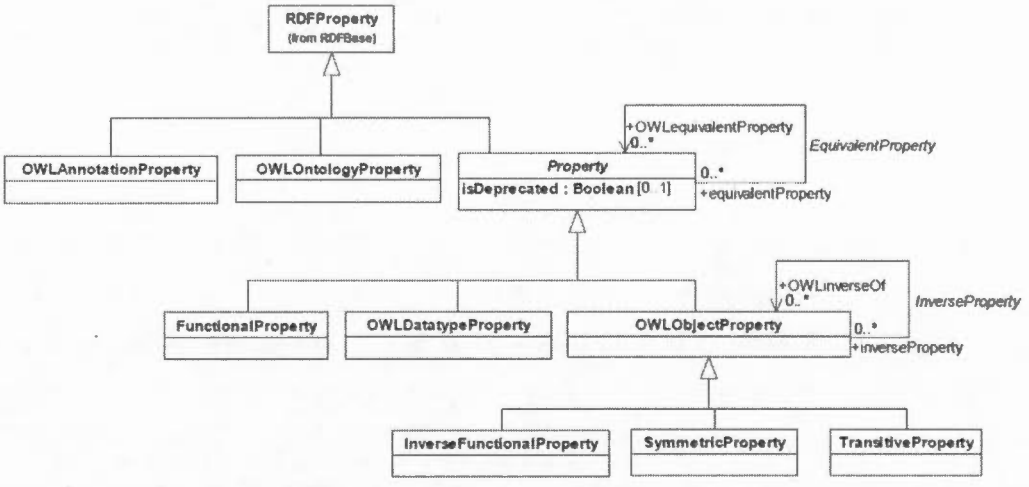


Figure .2 Diagramme des descriptions de propriétés OWL.

Mécanisme de traduction

Toutes les entités du méta-modèle OWL, citées précédemment, peuvent être codées et représentées au niveau d'une FCR en suivant le mécanisme suivant :

1. Chaque type d'entité *OWLClass* et *OWLObjectProperty* sera codé par un contexte formel. On notera contexte des *Classes* et contexte des *Rôles*, respectivement.
2. Les entités de type *OWLDatatypeProperty* et *FunctionalProperty* seront intégrées comme caractéristiques des *OWLClass*.
3. Les caractéristiques d'une entité *OWLClass* vont constituer les attributs du contexte formel des *Classes*.
4. Les entités de type *ComplementClass*, *IntersectionClass* et *UnionClass* seront représentées comme individus du contexte formel des *Classes*.
5. Les entités de type *InverseFunctionalProperty*, *SymmetricProperty*, *TransitiveProperty* ainsi que les *OWLRestrictions* seront intégrées comme caractéristiques des *OWLObjectProperty*.
6. Les caractéristiques d'une entité *OWLObjectProperty* vont constituer les attributs du contexte formel des *Rôles*.
7. Chaque type de liens inter-entités dans le méta-modèle sera représenté par une relation binaire entre deux contextes :
   - Le lien de type *InverseProperty* sera représenté par une relation binaire (contexte des *Rôles* X contexte des *Rôles*).
   - Les liens de types *EquivalentClass* et *DisjointClass* ainsi que les liens de types *ComplementClassForComplement*, *IntersectionClassForIntersection* et *UnionClassForUnion* seront représentés par des relations binaires (contexte des *Classes* X contexte des *Classes*).
   - Les autres types de liens inter-entités (entre *OWLClass* et *OWLObjectPro-*

perty) seront représentés par des relations binaires (contexte des *Classes* X contexte des *Rôles*) ou bien (contexte des *Rôles* X contexte des *Classes*).

ANNEXE 2

Classification des mesures de similarité sémantique

Les auteurs dans (Slimani *et al.*, 2007) proposent une classification des techniques existantes pour mesurer la similarité sémantique en quatre grandes familles :

Approches basées sur les arcs

Les mesures appartenant à cette famille se basent sur la structure hiérarchique des éléments dans l'ontologie pour calculer la distance qui sépare ces éléments (Rada *et al.*, 1989; LEE *et al.*, 1993; Wu et Palmer, 1994; Valtchev et Euzenat, 1997; Ehrig *et al.*, 2005).

Approches basées sur les nœuds

Ces approches utilisent la notion d'entropie de la théorie d'information. La similarité entre deux concepts (noeuds) est mesurée par rapport à la quantité d'information qu'ils partagent. Cette quantité est calculée sur la base de la probabilité de l'utilisation d'une classe et de ses sous-classes (Resnik, 1995; Hirst et St-Onge, 1998; Lin, 1998).

Approches hybrides

Ces approches tentent de tirer profit des avantages des deux approches précédentes afin d'améliorer la pertinence des similarités calculées. Elles se basent sur

la distance entre les concepts et prennent en considération également le contenu informationnel de chaque concept (Jiang et Conrath, 1997; Leacock et Chodorow, 1998).

Approches basées sur l'espace vectoriel

Les approches appartenant à cette classe (ex. *indice de Jaccard*, *mesure de Cosine*, *distance Euclidienne*, *indice de Dice*) commencent par représenter chaque concept par un vecteur dans un espace multi-dimensionnel. Puis, calculer la similarité entre deux concepts en se basant sur une certaine mesure (ex. *mesure de cosine*, *distance euclidienne*, etc.) entre les deux vecteurs relatifs.

ANNEXE 3

Détails des résultats de l'étude expérimentale

Résultats de l'expérimentation 4 pour le scénario 1

Ontologie	VP	FP	VN	FN	Précision	Rappel	F-mesure
CMS	5	2	0	1	0.71	0.83	0.76
Travel	30	7	0	4	0.81	0.88	0.84
Factbook-ont	40	14	0	4	0.74	0.90	0.81
Univ-cs	48	11	0	5	0.81	0.90	0.85
People	55	28	0	5	0.66	0.91	0.76
Soft-onto	60	25	0	6	0.70	0.90	0.79
Tourism	70	42	2	6	0.62	0.92	0.74
Ka	88	23	1	8	0.79	0.91	0.85
Pizza	91	42	1	9	0.68	0.91	0.78

Résultats de l'expérimentation 4 pour le scénario 2

Ontologie	VP	FP	VN	FN	Précision	Rappel	F-mesure
CMS	4	0	2	2	1	0.66	0.80
Travel	32	6	1	2	0.84	0.94	0.88
Factbook-ont	40	6	1	4	0.86	0.90	0.88
Univ-cs	48	11	0	5	0.81	0.90	0.85
People	56	10	2	4	0.84	0.93	0.88
Soft-onto	60	12	2	6	0.83	0.90	0.86
Tourism	70	28	9	6	0.71	0.92	0.80
Ka	90	23	0	6	0.79	0.93	0.86
Pizza	94	32	5	6	0.74	0.94	0.83

Résultats de l'expérimentation 4 pour le scénario 3

Ontologie	VP	FP	VN	FN	Précision	Rappel	F-mesure
CMS	6	2	0	0	0.75	1	0.85
Travel	34	0	7	0	1	1	1
Factbook-ont	44	5	2	0	0.89	1	0.94
Univ-cs	52	7	4	1	0.88	0.98	0.92
People	58	1	11	2	0.98	0.96	0.97
Soft-onto	63	8	6	3	0.88	0.95	0.91
Tourism	73	5	32	3	0.93	0.96	0.94
Ka	93	13	10	3	0.87	0.96	0.92
Pizza	96	22	15	4	0.81	0.96	0.88

BIBLIOGRAPHIE

Alani, H. et Brewster, C. (2006). Metrics for ranking ontologies. Dans *4th International Workshop on Evaluation of Ontologies for the Web at the 15th International World Wide Web Conference (WWW'2006)*.

Alfonseca, E. et Manandhar, S. (2002). Extending a lexical ontology by a combination of distributional semantics signatures. Dans *Knowledge Engineering and Knowledge Management : Ontologies and the Semantic Web*, 1–7. Springer.

Aussenac-Gilles, N., Despres, S. et Szulman, S. (2008). The terminae method and platform for ontology engineering from texts. *Bridging the Gap between Text and Knowledge-Selected Contributions to Ontology Learning and Population from Text*, 199–223.

Azpírez, J., Gómez-Pérez, A., Lozano-Tello, A. et Pinto, S. (1998). (onto) 2 agent : an ontology-based www broker to select ontologies.

Bain, M. (2003). Inductive construction of ontologies from formal concept analysis. In *AI 2003 : Advances in Artificial Intelligence* 88–99. Springer.

Barbut, M. et Monjardet, B. (1970). *Ordre et classification : algèbre et combinatoire*, volume 2. Hachette Paris.

Baumeister, J. et Seipel, D. (2005). Smelly owls-design anomalies in ontologies. Dans *FLAIRS Conference*, 215–220.

Baumeister, J. et Seipel, D. (2006). Verification and refactoring of ontologies with rules. Dans *Managing Knowledge in a World of Networks*, 82–95. Springer.

Behkamal, B., Naghibzadeh, M. et Moghadam, R. A. (2010). Using pattern detection techniques and refactoring to improve the performance of asmov. Dans *the 5th International Symposium on Telecommunications (IST'2010)*, 979–984. IEEE.

Bendaoud, R., Napoli, A. et Toussaint, Y. (2008). Formal concept analysis : A unified framework for building and refining ontologies. Dans *Knowledge Engineering : Practice and Patterns*, 156–171. Springer.

Biemann, C. (2005). Ontology learning from text : A survey of methods. Dans *LDV forum*, volume 20, 75–93.

Birkhoff, G. (1967). *Lattice theory*, volume 25. American Mathematical Society Colloquium Publications.

Blázquez, J., Fernández, M., García-Pinar, J. M. et Gómez-Pérez, A. (1998). Building ontologies at the knowledge level using the ontology design environment.

Bordat, J. P. (1986). Calcul pratique du treillis de galois d'une correspondance. *Mathématiques et Sciences humaines*, *96*, 31–47.

Borlund, P. (2003). The concept of relevance in ir. *Journal of the American Society for information Science and Technology*, *54*(10), 913–925.

Borst, W. N. (1997). *Construction of engineering ontologies for knowledge sharing and reuse.* PhD Thesis, University of Twente, Enschede, NL.

Budanitsky, A. et Hirst, G. (2006). Evaluating wordnet-based measures of semantic distance. *Computational Linguistics*, *32*(1), 13–47.

Buitelaar, P., Cimiano, P. et Magnini, B. (2005). *Ontology Learning from Text : Methods, Evaluation and Applications*, volume 123. IOS Press.

Carpineto, C. et Romano, G. (2004). *Concept data analysis : Theory and applications.* Wiley.

Chalupsky, H. (2000). Ontomorph : A translation system for symbolic knowledge. Dans *KR*, 471–482.

Cimiano, P., Hotho, A. et Staab, S. (2005). Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research (JAIR)*, *24*, 305–339.

Cimiano, P., Hotho, A., Stumme, G. et Tane, J. (2004). Conceptual knowledge processing with formal concept analysis and ontologies. Dans *Concept Lattices*, 189–207. Springer.

Clemente, J., Ramírez, J. et De Antonio, A. (2011). A proposal for student modeling based on ontologies and diagnosis rules. *Expert Systems with Applications*, *38*(7), 8066–8078.

Costa, M., Reeve, S., Grumbling, G. et Osumi-Sutherland, D. (2013). The drosophila anatomy ontology. *J. Biomedical Semantics*, *4*, 32.

Dao, M. (2003). Validation sur de grands projets, projet macao (rntl). *rapport no sous projet MACAO*, *5*.

Dao, M., Huchard, M., Rouane-Hacene, M., Roume, C. et Valtchev, P. (2004). Improving generalization level in uml models iterative cross generalization in practice. Dans *Conceptual Structures at Work*, 346–360. Springer.

Davey, B. A. et Priestley, H. A. (1990). *Introduction to Lattices and Order*. Cambridge University Press.

Dieng, R., Corby, O., Gandon, F., Giboin, A., Golebiowska, J., Matta, N. et Ribière, M. (2001). *Méthodes et outils pour la gestion des connaissances : une approche pluridisciplinaire du" Knowlegde Management"*, volume 2. Dunod.

Domingue, J. (1998). Tadzebao and webonto : Discussing, browsing, and editing ontologies on the web.

Drumond, L. et Girardi, R. (2008). A survey of ontology learning procedures. *The 3rd Workshop on Ontologies and their Applications (WONTO)*, *427*.

Duineveld, A., Stoter, R., Weiden, M., Kenepa, B. et Benjamins, V. (2000). Wondertools? a comparative study of ontological engineering tools. *International Journal of Human-Computer Studies*, *52*(6), 1111–1133.

Ehrig, M., Haase, P., Hefke, M. et Stojanovic, N. (2005). Similarity for ontologies-a comprehensive framework. *the 13th European Conference on Information Systems (ECIS)*.

Euzenat, J. et Valtchev, P. (2003). An integrative proximity measure for ontology alignment. Dans *the Semantic Integration workshop at the International Semantic Web Conference (ISWC)*, 33–38.

Euzenat, J. et Valtchev, P. (2004). Similarity-based ontology alignment in owl-lite. Dans *European Conference on Artificial Intelligence (ECAI)*, volume 16.

Farquhar, A., Fikes, R. et Rice, J. (1997). The ontolingua server : A tool for collaborative ontology construction. *International journal of human-computer studies*, *46*(6), 707–727.

Fernández-López, M. (1999). Overview of methodologies for building ontologies.

Fernández-López, M., Gómez-Pérez, A. et Juristo, N. (1997). Methontology : from ontological art towards ontological engineering.

Formica, A. (2006). Ontology-based concept similarity in formal concept analysis. *Information Sciences*, *176*(18), 2624–2641.

Fowler, M. (1999). *Refactoring : improving the design of existing code*. Pearson Education India.

Fürst, F. (2002). L'ingénierie ontologique. *Rapport de recherche, Institut de Recherche en Informatique de Nantes.*

Gailly, F. et Poels, G. (2007). Ontology-driven business modelling : improving the conceptual representation of the rea ontology. Dans *Conceptual Modeling (ER'2007)*, 407–422. Springer.

Ganter, B. (1984). Two basic algorithms in concept analysis. *Preprint 831, Technische Hochschule Darmstadt.*

Ganter, B. et Wille, R. (1999). *Formal Concept Analysis : Mathematical Foundations.* Springer.

Godin, R. et Mili, H. (1993). Building and maintaining analysis-level class hierarchies using galois lattices. Dans *ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'93)*, volume 28, 394–410. ACM.

Godin, R., Mili, H., Mineau, G. W., Missaoui, R., Arfi, A. et Chau, T.-T. (1998). Design of class hierarchies based on concept (galois) lattices. *TAPOS*, *4*(2), 117–134.

Godin, R., Missaoui, R. et Alaoui, H. (1995). Incremental concept formation algorithms based on galois (concept) lattices. *Computational intelligence*, *11*(2), 246–267.

Godin, R. et Valtchev, P. (2005). Formal concept analysis-based class hierarchy design in object-oriented software development. Dans *Formal Concept Analysis*, 304–323. Springer.

Gómez-Pérez, A. (1998). Knowledge sharing and reuse. *Handbook of applied expert systems*, 10–11.

Gómez-Pérez, A. (1999). Ontological engineering : A state of the art. *Expert Update : Knowledge Based Systems and Applied Artificial Intelligence*, *2*(3), 33–43.

Gómez-Pérez, A. (2001). Evaluation of ontologies. *International Journal of intelligent systems*, *16*(3), 391–409.

Gröner, G. et Staab, S. (2010). Categorization and recognition of ontology refactoring pattern. *Technical Report 09/2010.*

Gruber, T. R. *et al.* (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, *5*(2), 199–220.

Grüninger, M. et Fox, M. S. (1995). Methodology for the design and evaluation of ontologies.

Guarino, N. (1998). *Formal ontology in information systems.* Formal Ontology in Information Systems (FOIS'98), Trento, Italy, IOS Press.

Haase, P., Motta, E. et Studer, R. (2008). Infrastructure for semantic applications-neon toolkit goes open source. *ERCIM News*, *2008*(72).

Hirst, G. et St-Onge, D. (1998). Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet : An electronic lexical database*, *305*, 305–332.

Hovy, E. (1998). Combining and standardizing large-scale, practical ontologies for machine translation and other uses. Dans *the 1st International Conference on Language Resources and Evaluation (LREC)*, 535–542.

Jiang, J. J. et Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *the 10th International Conference on Research on Computational Linguistics.*

Kassel, G. (2002). Ontospec : une méthode de spécification semi-informelle d'ontologies. *Actes des 13e journées francophones d'Ingénierie des Connaissances*, 75–87.

Katsiouli, P. (2007). *Ontology Learning from Text : Methods & Tools.*

Kehagias, D. D., Kontotasiou, D. et Tzovaras, D. (2010). Evaluation framework for ontology development and management methodologies. *Ontology Repositories and Editors for the Semantic Web (ORES'2010)*, 134–145.

Kehagias, D. D., Papadimitriou, I., Hois, J., Tzovaras, D. et Bateman, J. (2008). A methodological approach for ontology evaluation and refinement. Dans *ASK-IT International Conference.*

Kuznetsov, S., Obiedkov, S. et Roth, C. (2007). Reducing the representation complexity of lattice-based taxonomies. Dans *Conceptual Structures : Knowledge Architectures for Smart Applications*, 241–254. Springer.

Kuznetsov, S. O. (2007). On stability of a formal concept. *Annals of Mathematics and Artificial Intelligence*, *49*(1-4), 101–115.

Lamsfus, C., Alzua-Sorzabal, A., Martin, D., Salvador, Z. et Usandizaga, A. (2009). Human-centric ontology-based context modelling in tourism. Dans *the International Conference on Knowledge Engineering and Ontology Development (KEOD'2009)*, 424–434.

Leacock, C. et Chodorow, M. (1998). Combining local context and wordnet similarity for word sense identification. *WordNet : An electronic lexical database*, *49*(2), 265–283.

LEE, J. H., KIM, M. H. et LEE, Y. J. (1993). Information retrieval based on conceptual distance in is-a hierarchies. *Journal of documentation*, *49*(2), 188–207.

Lin, D. (1998). An information-theoretic definition of similarity. Dans *ICML*, volume 98, 296–304.

Maedche, A., Schnurr, H.-P., Staab, S. et Studer, R. (2000). Representation language-neutral modeling of ontologies. Dans *the German Workshop "Modellierung-2000". Koblenz, Germany*, 129–142. Citeseer.

Maedche, A. et Staab, S. (2000). The text-to-onto ontology learning environment. Dans *the Eight International Conference on Conceptual Structures*, volume 38.

Mazuel, L. et Sabouret, N. (2007). Degré de relation sémantique dans une ontologie pour la commande en langue naturelle. Dans *Ingénierie des Connaissances (IC2007)*, 1–10.

Mazuel, L. et Sabouret, N. (2008). Protocole d'évaluation d'une mesure de degré de relation sémantique. *Atelier Mesures de Similarité Sémantique, à EGC*, 77–86.

McGuinness, D. L., Fikes, R., Rice, J. et Wilder, S. (2000). An environment for merging and testing large ontologies. Dans *KR*, 483–493.

Michalski, R. S., Carbonell, J. G. et Mitchell, T. M. (2013). *Machine learning : An artificial intelligence approach*. Springer Science & Business Media.

Mizoguchi, R. et Bourdeau, J. (2004). Le rôle de l'ingénierie ontologique dans le domaine des eiah. *Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF)*, *11*, 231–246.

Mizoguchi, R. et Ikeda, M. (1998). Towards ontology engineering. *Journal-Japanese Society for Artificial Intelligence*, *13*, 9–10.

Montes-y Gómez, M., Gelbukh, A. et López-López, A. (2000). Comparison of conceptual graphs. Dans *the 1st Mexican International Conference on Artificial Intelligence (MICAI'2000)*, 548–556. Springer.

Myaeng, S. H. et López-López, A. (1992). Conceptual graph matching : a flexible algorithm and experiments. *Journal of Experimental & Theoretical Artificial Intelligence*, *4*(2), 107–126.

Nanda, J., Simpson, T. W., Kumara, S. R. et Shooter, S. B. (2006). A methodology for product family ontology development using formal concept analysis and web ontology language. *Journal of computing and information science in engineering*, *6*, 103–113.

Nourine, L. et Raynaud, O. (1999). A fast algorithm for building lattices. *Information processing letters*, *71*(5), 199–204.

Noy, N. F. et Musen, M. A. (2000). Algorithm and tool for automated ontology merging and alignment. Dans *the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831.*

Ostrowski, D. A. (2008). Ontology refactoring. Dans *IEEE International Conference on Semantic Computing*, 476–479. IEEE.

Paşca, M. (2005). Finding instance names and alternative glosses on the web : Wordnet reloaded. Dans *Computational Linguistics and Intelligent Text Processing*, 280–292. Springer.

Poesio, M. et Almuhareb, A. (2005). Identifying concept attributes using a classifier. Dans *the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, 18–27. Association for Computational Linguistics.

Poveda Villalon, M., Suárez-Figueroa, M. C., García-Castro, R. et Gómez-Pérez, A. (2010). A context ontology for mobile environments. *Workshop on Context, Information and Ontologies (CIAO'2010) co-located with EKAW'2010.*

Rada, R., Mili, H., Bicknell, E. et Blettner, M. (1989). Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, *19*(1), 17–30.

Rahm, E. et Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *the International Journal on Very Large Data Bases (VLDB)*, *10*(4), 334–350.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. *the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, 448–453.

Resnik, P. (1999). Semantic similarity in a taxonomy : An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research (JAIR)*, *11*, 95–130.

Rijsbergen, C. (1975). Information retrieval. Butterworths.

Roth, C., Obiedkov, S. et Kourie, D. (2008). Towards concise representation

for taxonomies of epistemic communities. Dans *Concept Lattices and their Applications*, 240–255. Springer.

Rouane-Hacene, M., Dao, M., Huchard, M., Valtchev, P. *et al.* (2007). Analyse formelle de données relationnelles poùr la réingénierie des modèles uml. Dans *Langages et Modèles à Objets (LMO'2007)*, 151–166.

Rouane-Hacene, M., Fennouh, S., Nkambou, R. et Valtchev, P. (2010). Refactoring of ontologies : Improving the design of ontological models with concept analysis. Dans *IEEE International Conference on Tools with Artificial Intelligence (ICTAI'2010)*, volume 2, 167–172. IEEE.

Rouane-Hacene, M., Huchard, M., Napoli, A. et Valtchev, P. (2013). Relational concept analysis : mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, 1–28.

Rouane-Hacene, M., Napoli, A., Valtchev, P., Toussaint, Y. et Bendaoud, R. (2008). Ontology learning from text using relational concept analysis. Dans *International MCETECH Conference on e-Technologies*, 154–163. IEEE.

Rouane-Hacene, M., Valtchev, P. et Nkambou, R. (2011). Supporting ontology design through large-scale fca-based ontology restructuring. Dans *Conceptual Structures for Discovering Knowledge*, 257–269. Springer.

Schamber, L., Eisenberg, M. B. et Nilan, M. S. (1990). A re-examination of relevance : toward a dynamic, situational definition. *Information processing & management*, *26*(6), 755–776.

Seco, N., Veale, T. et Hayes, J. (2004). An intrinsic information content metric for semantic similarity in wordnet. Dans *the 16th European Conference on Artificial Intelligence (ECAI)*, volume 16, 1089–1090.

Slimani, T., BenYaghlane, B. et Mellouli, K. (2007). Une extension de mesure de similarité entre les concepts d ?une ontologie. Dans *International conference on Sciences of Electronic, Technologies of Information and Telecommunications (SETIT)*, 1–10.

Snelting, G. (2000). Software reengineering based on concept lattices. Dans *the Fourth European Conference on Software Maintenance and Reengineering*, 3–10. IEEE.

Sowa, J. F. (1983). Conceptual structures : information processing in mind and machine.

Staab, S., Studer, R., Schnurr, H.-P. et Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent systems*, *16*(1), 26–34.

Studer, R., Benjamins, V. R. et Fensel, D. (1998). Knowledge engineering : principles and methods. *Data & knowledge engineering*, *25*(1), 161–197.

Stumme, G. (2005). Ontology merging with formal concept analysis. *Semantic Interoperability and Integration*, (04391).

Stumme, G. et Maedche, A. (2001). Fca-merge : Bottom-up merging of ontologies. Dans *IJCAI*, volume 1, 225–230.

Suarez-Figueroa, M. C. et Gomez-Perez, A. (2008). First attempt towards a standard glossary of ontology engineering terminology.

Suarez-Figueroa, M. C., Gomez-Perez, A. et Fernandez-Lopez, M. (2012). The neon methodology for ontology engineering. Dans *Ontology engineering in a networked world*, 9–34. Springer.

Šváb-Zamazal, O., Svátek, V., Meilicke, C. et Stuckenschmidt, H. (2008). Testing the impact of pattern-based ontology refactoring on ontology matching results. Dans *the 7th International Semantic Web Conference*, 240–271. Citeseer.

Thieu, M., Steichen, O., Zapletal, E., Jaulent, M.-C. et Le Bozec, C. (2004). Mesures de similarité pour l'aide au consensus en anatomie pathologique. Dans *15èmes Journées francophones d'Ingénierie des Connaissances*, 225–236. Presses universitaires de Grenoble.

Torniai, C., Essaid, S., Lowe, B., Corson-Rikert, J. et Haendel, M. (2013). Finding common ground : integrating the eagle-i and vivo ontologies. Dans *ICBO*, 46–49.

Uschold, M. et King, M. (1995). *Towards a methodology for building ontologies*. Citeseer.

Valtchev, P. (1999). An algorithm for minimal insertion in a type lattice. *Computational intelligence*, *15*(1), 63–78.

Valtchev, P. et Euzenat, J. (1997). Dissimilarity measure for collections of objects and values. Dans *Lecture Notes in Computer Science*, volume 1280, 259–272. Springer.

Valtchev, P., Missaoui, R. et Lebrun, P. (2002). A partition-based approach towards constructing galois (concept) lattices. *Discrete Mathematics*, *256*(3), 801–829.

Welty, C. et Guarino, N. (2001). Supporting ontological analysis of taxonomic relationships. *Data & Knowledge Engineering*, *39*(1), 51–74.

Widdows, D. (2003). Unsupervised methods for developing taxonomies by combining syntactic and statistical information. Dans *Human Langauge Technology/North American Chapter of the Association for Computational Linguistics*, 197–204. Association for Computational Linguistics.

Wimmer, M. (2009). A meta-framework for generating ontologies from legacy schemas. Dans *the 20th International Workshop on Database and Expert Systems Application (DEXA'2009)*, 474–479. IEEE.

Witschel, H. F. (2005). Using decision trees and text mining techniques for extending taxonomies. Dans *the Workshop on Learning and Extending Lexical Ontologies by Using Machine Learning Methods.*

Wu, Z. et Palmer, M. (1994). Verbs semantics and lexical selection. Dans *the 32nd annual meeting on Association for Computational Linguistics*, 133–138. Association for Computational Linguistics.

Yang, D. et Powers, D. M. (2005). Measuring semantic similarity in the taxonomy of wordnet. Dans *the Twenty-eighth Australasian conference on Computer Science*, volume 38, 315–322. Australian Computer Society, Inc.

Zouaq, A. et Nkambou, R. (2008). Building domain ontologies from text for educational purposes. *IEEE Transactions on Learning Technologies*, *1*(1), 49–62.