Représenter, opérationnaliser, aligner et évaluer des Ontologies Denses : une approche et un outil fondés sur le modèle des Graphes Conceptuels

Frédéric Fürst*, Francky Trichet**

* LARIA - Laboratoire de Recherche en Informatique d'Amiens (CNRS-FRE 2733)
UPJV, 33 rue Saint Leu - 80039 Amiens Cedex 01
frederic.furst@u-picardie.fr
** LINA - Laboratoire d'Informatique de Nantes Atlantique (CNRS-FRE 2729)
Equipe COD - Connaissances & Décision
2 rue de la Houssinière BP 92208 - 44322 Nantes Cedex 03
francky.trichet@univ-nantes.fr

Résumé. À l'heure actuelle, les ontologies sont au cœur de nombreuses applications car, outre le fait d'établir un consensus sur le vocabulaire conceptuel d'un domaine, elles permettent également de raisonner sur des assertions de ce domaine. Néanmoins, dans de nombreux contextes et en particulier dans le cadre du Web Sémantique dont le défi est d'offrir la possibilité d'automatiser la mise en œuvre de raisonnements, il s'avère de plus en plus indispensable de considérer des *ontologies denses*, *i.e.* des ontologies intégrant l'ensemble des axiomes permettant de fixer toute la sémantique du domaine considéré, en comparaison aux *ontologies* dites *légères* qui, elles, n'incluent pas d'axiomes et sont uniquement fondées sur des hiérarchies de concepts et de relations, éventuellement enrichies de propriétés classiques telles que les propriétés algébriques des relations ou l'exclusion de deux concepts (cf. l'expressivité du langage OWL). Peu de travaux relevant de l'ingénierie des ontologies prennent en compte la représentation explicite des axiomes ne pouvant s'exprimer à l'aide de ces propriétés classiques. Or, la représentation formelle de ce type d'axiomes (et leur prise en compte au sein de mécanismes de raisonnement) s'avère cruciale dans de nombreuses applications. Cet article présente un atelier d'ingénierie des ontologies denses, basé sur le modèle des Graphes Conceptuels. Cet atelier, appelé TooCoM, met l'accent sur la prise en compte des axiomes, tant d'un point de vue modélisation et représentation des connaissances que d'un point de vue mise en œuvre de raisonnements à des fins d'opérationnalisation des connaissances (*i.e.* intégration d'une ontologie au sein d'un Système à Base de Connaissances), d'évaluation (*i.e.* vérification et validation) et d'alignement d'ontologies (*i.e.* interopérabilité sémantique).

1 Introduction

Les ontologies sont actuellement au cœur de nombreuses applications, en particulier le Web Sémantique, car elles ont pour objectif de supporter la gestion des connaissances et le raisonnement sur ces connaissances, dans une optique d'interopérabilité sémantique entre agents humains et/ou artificiels. Néanmoins, la plupart des travaux actuels relevant de l'ingénierie des ontologies se limitent à la construction d'ontologies qualifiées de légères (*lightweight ontologies*) car composées simplement d'une hiérarchie de concepts (éventuellement enrichie de propriétés telles que l'exclusion ou l'abstraction) parfois associée à une hiérarchie de relations (éventuellement enrichie de propriétés algébriques). Allégées en sémantique, ces ontologies ne permettent pas de prendre en compte toutes les connaissances d'un domaine donné, en particulier les règles et les contraintes qui fixent l'interprétation des concepts et relations le caractérisant [20]. Ce déficit de sémantique s'illustre dans la faible expressivité du langage OWL, ce standard étant focalisé sur la représentation d'ontologies légères et délaissant les connaissances liées à l'inférence (principalement règles et contraintes). Ceci limite fortement son utilisation, tant pour représenter des connaissances inférentielles que pour la mise en œuvre de ce type de connaissances au sein des raisonnements sous-jacents aux activités de vérification, validation, alignement et fusion d'ontologies mais également aux activités liées à l'utilisation effective des ontologies à des fins, par exemple, de recherche d'information par exploitation d'annotations formelles de contenus.

L'objectif de cet article est double. Tout d'abord, il s'agit de souligner l'enjeu actuel de développer des travaux dédiés à l'ingénierie des ontologies denses sémantiquement parlant (*heavyweight ontologies*), *i.e.* des ontologies qui en plus d'intégrer les concepts et relations (structurés au sein de hiérarchies fondées sur la relation de Spécialisation/Généralisation) du domaine considéré, incluent les axiomes (règles et contraintes) qui régissent ce domaine. La prise en compte de ce niveau axiomatique, qui s'avère indispensable dans de nombreux domaines tels que celui étudié dans nos travaux et relatif à l'analyse de textes réglementaires appliqués au domaine Hygiène Sécurité et Environnement, permet d'envisager la mise en œuvre de raisonnements automatiques, véritable clé de voûte de la popularisation du Web Sémantique : "*For the semantic web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning*" T. Berners-Lee [4]. L'engouement actuel autour de la standardisation d'un langage de règles pour le Web Sémantique, tel que SWRL, illustre parfaitement ce besoin.

Dans un second temps, il s'agit de montrer que proposer un atelier fondé sur le modèle des Graphes Conceptuels (GC) s'avère être une solution adaptée pour gérer des ontologies denses, étant donné que les axiomes, éléments intrinsèques des ontologies denses, sont naturellement représentables en termes de graphes et facilitement exploitables à l'aide d'opérations de graphes. Notre contribution à l'ingénierie des ontologies denses repose sur les éléments suivants : (1) le langage OCGL (*Ontology Conceptual Graphs Language*) fondé sur le modèle des Graphes Conceptuels et adapté à la représentation du niveau axiomatique d'une ontologie dense, (2) une méthode d'opérationnalisation associée à OCGL qui permet, en fonction du contexte applicatif visé, d'intégrer automatiquement une ontologie dense au sein d'un Système à Base de Connaissances (et par la même de la doter d'une sémantique opérationnelle complémentaire de sa sémantique formelle) et enfin (3) une méthode d'alignement intentionnelle qui repose principalement sur la comparaison structurelle des axiomes (et par conséquent de la sémantique qu'ils véhiculent) pour identifier et évaluer des liens d'identités entre les

concepts et relations de deux ontologies supposées bâties sur des domaines connexes. Tous ces éléments sont implémentés au sein de l'atelier TooCoM (*a Tool to Operationalize an Ontology with the Conceptual Graph Model*) disponible sous licence GNU GPL sur SourceForge : `http ://sourceforge.net/projects/toocom/`.

La suite de cet article est structurée comme suit. La section 2 introduit les composants et principes sur lesquels sont fondés nos travaux de représentation et de raisonnement sur les ontologies denses, à savoir (1) le modèle des Graphes Conceptuels, (2) le langage OCGL de représentation d'ontologies denses, (3) l'ontologie de représentation MetaOCGL[1] et (4) les fondements du processus d'opérationnalisation d'une ontologie. La section 3 présente notre méthode d'alignement d'ontologies denses (fondée principalement sur la comparaison topologique des axiomes représentés à l'aide de graphes) et introduit les premiers résultats expérimentaux.

2 Représenter et opérationnaliser des ontologies denses

2.1 Contexte des travaux : le modèle des Graphes Conceptuels

Le modèle des Graphes Conceptuels (GC), introduit par J. SOWA [19], est un modèle opérationnel de représentation de connaissances, qui appartient à la famille des réseaux sémantiques. Ce modèle est mathématiquement fondé sur la logique et la théorie des graphes [19]. Cependant, pour raisonner à l'aide de GC, deux approches peuvent être distinguées : (1) considérer les GC comme une interface graphique pour la logique et donc raisonner à l'aide de la logique et (2) considérer les GC comme un modèle de représentation à part entière disposant de ses propres mécanismes de raisonnement fondés sur la théorie des graphes. Dans le cadre de nos travaux qui reposent sur l'utilisation de la famille *SG* [1] (incluant les Graphes Conceptuels Simples, les Règles de GC et les Contraintes Positives et Négatives de GC), nous adoptons la seconde approche en utilisant la *projection* (une opération de graphes correspondant à un homomorphisme) comme opérateur de raisonnement ; la *projection* est complète et cohérente vis-à-vis de la déduction en logique du premier ordre [6].

Le choix du modèle des GCs comme modèle de base au coeur de TooCoM (notre atelier d'ingénierie des ontologies denses) est motivé par différents aspects : (1) la représentation graphique inhérente au modèle facilite la représentation et la visualisation des composants d'une ontologie dense (en particulier les axiomes), (2) les raisonnements de la famille SG sont basés sur des opérations de graphes (sans transformation logique), ce qui permet de produire des traces d'inférences plus lisibles pour l'ingénieur de la connaissance (car exprimées graphiquement) et ainsi facilite son travail de validation de l'ontologie dense en cours de construction,

[1] Comme introduit dans [15], il existe différents types d'ontologies : (1) *ontologies de représentation*, définissant les primitives d'un paradigme de représentation des connaissances (par exemple la Frame Ontology pour le paradigme des frames [16]), (2) *ontologies de haut niveau*, définissant des notions abstraites et/ou de sens commun telles que le temps, l'espace, les quantités (par exemple SUO, Standard Upper Ontology proposée par un groupe de travail IEEE - `http ://suo.ieee.org`), (3) *ontologies linguistiques*, définissant des ensembles d'unités lexicales liées par des relations linguistiques telles que la synonymie ou l'hyperonymie (par exemple Wordnet - `http ://www.wordnet.princeton.edu/`), (4) *ontologies de domaine*, définissant les connaissances d'un domaine borné (par exemple UMLS, Unified Medical Language System, pour le domaine médical - `http ://www.nlm.nih.gov/research/umls/`) et (5) *ontologies de PSM - Problem Solving Method*, définissant des méthodes de résolution de problèmes génériques (par exemple les tâches et méthodes de Chandrasekaran [5]).

(3) les algorithmes développés pour la famille SG utilisent des techniques de combinatoires (issues en particulier de la théorie des graphes et des problèmes de satisfaction de contraintes) qui les rendent particulièrement efficaces et enfin (4) la famille SG est implémentée au sein de COGITANT [14], une API de développement GPL sur laquelle est développé TooCom.

2.2 OCGL : un langage de représentation d'ontologies denses

Le langage OCGL (*Ontology Conceptual Graphs Language* [10]) que nous proposons pour représenter une ontologie (au niveau conceptuel) est fondé sur trois primitives : *Concept*, *Relation* et *Axiome*. Représenter une ontologie en OCGL consiste principalement à (1) spécifier le vocabulaire conceptuel du domaine considéré et (2) spécifier la sémantique de ce vocabulaire à l'aide d'axiomes.

Le vocabulaire conceptuel est composé d'un ensemble de **Concepts** et d'un ensemble de **Relations**. Ces deux ensembles peuvent être structurés soit en utilisant des propriétés conceptuelles bien connues que nous appelons *Schémas d'Axiomes*, soit en utilisant des *Axiomes de Domaine*. L'union des *Schémas d'Axiomes* et des *Axiomes de Domaine* correspond à ce que nous appelons les **Axiomes**.

– Les **Schémas d'Axiomes** proposés par défaut dans OCGL sont :

1. la relation *ISA* entre deux concepts ou deux relations (aussi appelée Subsomption ou relation de Spécialisation/Généralisation) et utilisée pour construire des taxinomies (arbres ou treillis de concepts et de relations) ;
2. la propriété d'*Abstraction* d'un concept (qui correspond à la notion d'*Exhaustive-Decomposition* dans certains travaux [15]). Un concept est dit abstrait s'il n'admet pas d'instance directe : toutes ses instances sont nécessairement instances d'un de ses concepts fils. L'abstraction implique donc l'hypothèse du monde clos. Par exemple, le concept $NombreEntier$ décomposé en $NombrePair$ et $NombreImpair$ est abstrait, car tout nombre est soit pair, soit impair ;
3. la propriété de *Disjonction* entre deux concepts. Notons qu'il est possible de définir une *Partition* [15] en utilisant conjointement l'abstraction et la disjonction. Par exemple, la décomposition de $NombreEntier$ en $NombrePair$ et $NombreImpair$ est une partition car $NombreEntier$ est un concept abstrait et $NombrePair$ et $NombreImpair$ sont disjoints ;
4. la *Signature* d'une relation (précisant les concepts liés par la relation considérée) ;
5. les *Propriétés Algébriques* d'une relation (symétrie, réflexivité, transitivité, irréflexivité, antisymétrie) ;
6. l'*Exclusivité* ou l'*Incompatibilité* de deux relations. L'incompatibilité de deux relations R_1 et R_2 est formalisée par $\neg(R_1 \wedge R_2)$: R_1 et R_2 ne peuvent être attestées simultanément sur les mêmes instances de concepts comme par exemple les relations $Mother(Woman, Human)$ et $Daughter(Woman, Human)$. L'exclusivité est formalisée par $\neg R_1 \Rightarrow R_2$: nier l'une des deux relations atteste nécessairement l'autre, par exemple $Higher(Human, Human)$ et $Smaller(Human, Human)$;
7. les *Cardinalités* (Maximale et Minimale) d'une relation.

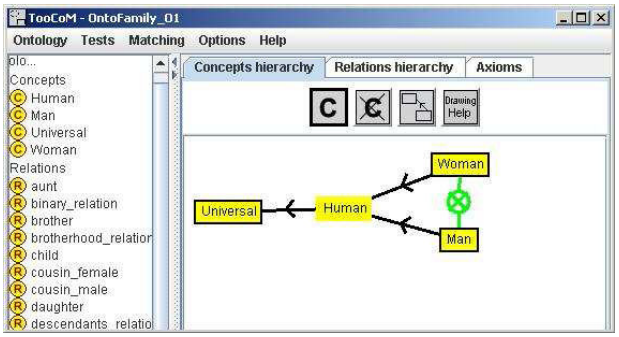

FIG. 1 – *La hiérarchie de concepts de l'ontologie OntoFamily O_1 dans TooCoM. Une flèche représente un lien de subsomption, un concept non encadré identifie un concept abstrait et les disjonctions entre concepts sont représentées par un cercle barré.*

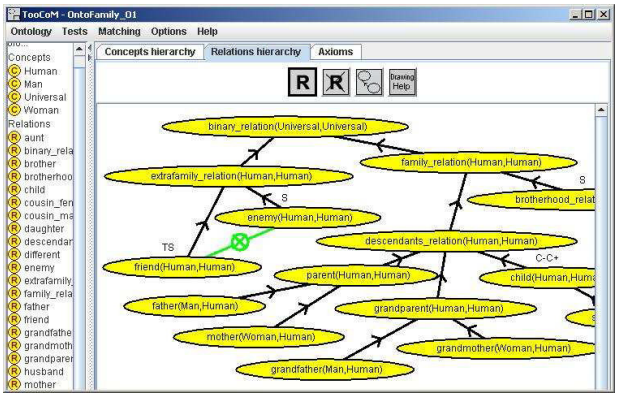

FIG. 2 – *Extrait de la hiérarchie de relations d'OntoFamily O_1 dans TooCoM. Un cercle barré représente une incompatibilité (ou une exclusivité) entre deux relations, les propriétés algébriques et les cardinalités sont symbolisées à l'aide de lettres (S pour la symétrie, T pour la transitivité, C+ et C- pour les cardinalités, etc.).*

– Les **Axiomes de Domaine** correspondent aux connaissances inférentielles du domaine qu'il n'est pas possible de représenter à l'aide des Schémas d'Axiomes, et qui donc ne correspondent pas à des propriétés classiques attestées sur les concepts ou sur les relations. La syntaxe graphique d'OCGL utilisée pour exprimer ces connaissances est fondée sur le modèle des Graphes Conceptuels. Ainsi, un Axiome de Domaine est composé d'un graphe *Antécédent* et d'un graphe *Conséquent*, la sémantique formelle d'une telle construction pouvant s'exprimer intuitivement comme suit : *« si le graphe Antécédent est attesté vrai, alors le graphe Conséquent est attesté vrai »*. La figure 3 présente le graphe OCGL correspondant à la connaissance inférentielle (fictive) « *Les ennemis de mes amis sont mes ennemis* ». Notons que cette assertion correspond bien à ce que nous appelons un *Axiome de Domaine* car il n'est pas possible de la représenter à l'aide d'un

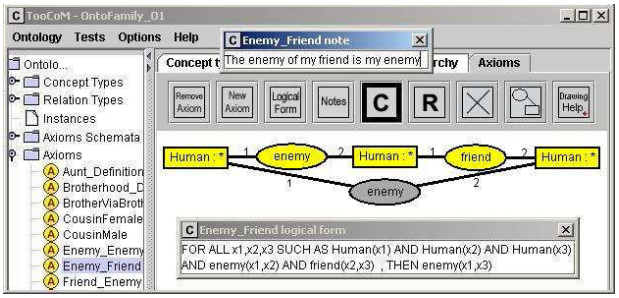

FIG. 3 – *Un axiome de OntoFamily dans TooCoM. Les nœuds clairs composent le graphe antécédent, les nœuds foncés le graphe conséquent. Chaque partie contient des nœuds concepts (rectangles) et des nœuds relations (ellipses). Un nœud concept est décrit par un label et un marqueur qui identifie l'instance considérée, sachant que le marqueur $*$ dénote une instance indéfinie. Un nœud relation est uniquement décrit par un label. Un arc entre un concept et une relation est étiqueté par la position du concept au sein de la signature de la relation. L'expression logique de l'axiome est automatiquement générée par TooCom.*

Schéma d'Axiome contrairement à l'assertion « *Les amis de mes amis sont mes amis* » qui elle est représentée à l'aide de la transitivité de la relation *Friend(Human,Human)*.

OCGL a été implémenté au sein de l'outil TooCoM[2] (*a Tool to Operationalize an Ontology with the Conceptual Graph Model*) dédié à l'édition et à l'opérationnalisation d'ontologies [9, 12]. Cet outil, fondé sur le modèle des Graphes Conceptuels et s'appuyant sur la plate-forme de manipulation de graphes conceptuels CoGITaNT [14], permet de définir les primitives conceptuelles (*i.e.* les Concepts et les Relations) et la sémantique formelle du domaine considéré (*i.e.* les Schémas d'Axiomes et Axiomes de Domaine) via une approche graphique. Les figures 1, 2 et 3 présentent respectivement la hiérarchie de concepts, un extrait de la hiérarchie de relations et un axiome de l'ontologie OntoFamily éditée sous TooCom et dédiée aux relations familiales[3].

Pour résumer, une *ontologie légère* est représentée en OCGL par le triplet {Concepts, Relations, Schémas d'Axiomes} et une *ontologie dense* par le quadruplet {Concepts, Relations, Schémas d'Axiomes, Axiomes de Domaine}. Il est important de souligner qu'il existe une correspondance partielle entre OCGL et OWL, et donc que TooCom permet d'importer et d'exporter des ontologies au format OWL en utilisant l'API d'OWL [2]. La plupart des primitives d'OWL possèdent une correspondance en OCGL. Néanmoins, les deux langages n'ont pas le même pouvoir d'expression. Ainsi, il n'est pas possible de représenter les Axiomes de Domaine en OWL (qui ne dispose pas encore d'un langage de règles), et OCGL ne permet pas de représenter certaines propriétés d'OWL telles que *allValuesFrom* ou *someValuesFrom*. De la même façon, OCGL se distingue de la famille *SG* en tant que langage de haut niveau dédié

[2] TooCoM est disponible en licence GNU GPL sur SourceForge : `http ://sourceforge.net/projects/toocom/`.

[3] Étant donné que l'ontologie construite a été définie en anglais, nous avons délibéremment choisi de traduire toutes nos figures en anglais de façon à conserver une unité de lecture des illustrations, et par conséquent de faciliter la compréhension des idées défendues. L'ontologie OntoFamily, bien que peu volumineuse, est intéressante car (1) elle couvre un domaine compréhensible par tous et (2) elle correspond parfaitement à une ontologie dense incluant Concepts, Relations, Schémas d'Axiomes et Axiomes de Domaines.

à la représentation d'ontologies denses. Certes, OCGL s'appuie sur les primitives de la SG en termes de représentation formelle mais il permet à l'ingénieur de la connaissance de s'abstraire des constructions de base des GCs (Règles GC et Contraintes GC), et donc de se placer à un niveau de représentation ontologique. Ainsi, l'ingénieur de la connaissance n'a pas à se préoccuper de la représentation opérationnelle d'une exclusion entre deux concepts en Graphes Conceptuels : il spécifie l'exclusion en OCGL et la traduction opérationnelle en primitives GC est automatique.

2.3 Opérationnaliser une ontologie dense

Une ontologie de domaine peut être utilisée (1) pour partager des connaissances entre agents humains et/ou artificiels, (2) pour raisonner sur des bases de connaissances ou (3) pour faciliter la recherche d'information. Afin de répondre à toutes ces finalités, une ontologie ne doit pas uniquement considérer le volet terminologique d'un domaine comme peut le faire un thésaurus, mais doit intégrer toutes les connaissances de ce dernier. Ainsi, les ontologies actuelles, qui correspondent pour la plupart à des ontologies qualifiées de "*légères*" (*lightweight ontologies*) car incluant seulement quelques propriétés telles que la subsomption et les propriétés algébriques, doivent évoluer vers des ontologies plus "*denses*" sémantiquement parlant (*heavyweight ontologies*) car incluant tous les axiomes permettant de représenter toute la sémantique du domaine considéré [20].

Néanmoins, afin de garantir l'indépendance des ontologies de domaine vis-à-vis des applications où elles sont utilisées, et donc leur réutilisabilité, la représentation des axiomes doit seulement préciser leur *sémantique formelle*, qui contraint l'interprétation des primitives conceptuelles (concepts et relations), sans fixer leur *sémantique opérationnelle*, qui spécifie la façon dont les axiomes sont utilisés pour raisonner [10]. Par exemple, l'axiome *« Les ennemis de mes ennemis sont mes amis »* peut être utilisé soit pour inférer une nouvelle situation (*i.e.* déduire automatiquement au sein d'une base d'assertions que lorsque qu'un ennemi d'un de mes ennemis est identifié alors ce dernier est un ami), soit pour vérifier la cohérence d'une situation (*i.e.* vérifier qu'il n'a pas été attesté au sein d'une base d'assertions que l'ennemi d'un de mes ennemis est mon ennemi). Par ailleurs, un axiome peut être implicitement (et automatiquement) appliqué par le système ou explicitement appliqué par l'utilisateur. La combinaison de ces deux critères, (i) utilisation pour déduire *vs* valider et (ii) utilisation implicite vs explicite, conduit à l'identification de quatre *Contextes d'Utilisation* d'un axiome :

1. *inférentiel implicite* permettant au système de produire automatiquement de nouvelles assertions à partir d'une base initiale ;
2. *inférentiel explicite* permettant à l'utilisateur final de produire de nouvelles assertions en mettant en œuvre la (ou les) connaissance(s) sous-jacente(s) à l'axiome ;
3. *validation implicite* permettant au système de vérifier automatiquement la cohérence de la base ;
4. *validation explicite* permettant à l'utilisateur de vérifier la base à sa demande[4].

[4]Notons que les deux aspects "déduction *vs* validation" et "implicite *vs* explicite" pris en compte pour décrire le contexte d'utilisation d'un axiome relèvent volontairement d'un niveau de granularité bas. Il est bien évidemment possible d'envisager la prise en compte de points de vue d'un plus haut niveau de description tels que le type de raisonnement envisagé (déduction, abduction, induction) ou l'objectif applicatif recherché (par exemple l'apprentissage interactif assisté ou la gestion d'une mémoire d'organisation).

L'utilisation d'une ontologie dense au sein d'une application (*i.e.* son intégration au sein d'un SBC opérationnel) nécessite une étape dite d'*opérationnalisation* [10] qui consiste (1) à spécifier la façon dont les axiomes seront utilisés à travers un *Scénario d'Usage* puis (2) à transcrire tous les axiomes sous une forme opérationnelle (dans notre cas, Règles GC et/ou Contraintes GC) en fonction du *Scénario d'Usage* retenu.

Un *Scénario d'Usage* décrit à quelles fins vont être utilisées les connaissances spécifiées dans l'ontologie, *i.e.* essentiellement à quoi vont servir les axiomes de l'ontologie. En effet, la représentation du vocabulaire conceptuel (concepts et relations) du domaine ne dépend pas des multiples contextes applicatifs possibles, la représentation d'un concept ou d'une relation étant la même dans le cas d'un système dédié à la validation de connaissances ou d'un système dédié à la production de connaissances. Seules les représentations opérationnelles des axiomes doivent être adaptées à l'objectif de l'application envisagée.

Étant donné que la sémantique opérationnelle de chaque axiome doit être spécifiée, construire un *Scénario d'Usage* consiste à choisir, pour chaque axiome, son *Contexte d'Utilisation*. En fonction de la structure de l'axiome et du contexte sélectionné, la forme opérationnelle produite peut correspondre soit à une Règle GC, soit à une Contrainte GC, soit à un ensemble de Règles et/ou de Contraintes (*cf.* [10] pour plus de détails sur ce processus de transformation automatique). Ainsi, comme le montre la figure 4, à partir d'une ontologie de domaine, il est possible de développer plusieurs ontologies opérationnelles, chacune de ces ontologies opérationnelles étant liée à un *Scénario d'Usage* différent (*i.e.* une combinaison unique des *Contextes d'Utilisation* des axiomes).

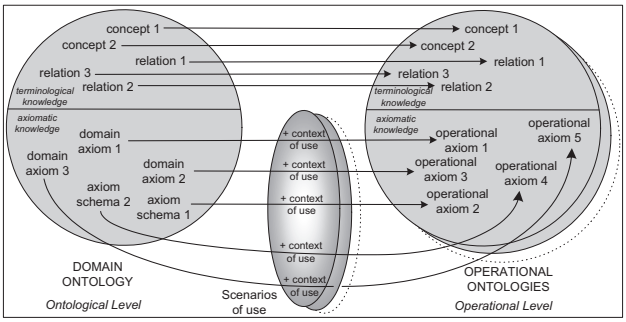

FIG. 4 – *Processus d'opérationnalisation d'une ontologie dense. Les connaissances terminologiques sont représentées de la même manière au niveau ontologique et au niveau opérationnel. La représentation des connaissances axiomatiques au niveau opérationnel dépend du scénario d'usage qui décrit la façon dont les axiomes sont utilisés pour raisonner sur l'ontologie opérationnelle. La représentation opérationnelle d'un axiome correspond à un ensemble de règles et/ou de contraintes.*

Opérationnaliser une ontologie de domaine correspond donc au développement d'un SBC capable de mettre en œuvre des raisonnements sur une base de faits. Par exemple, opérationnaliser OntoFamily dans le cadre d'un scénario de type inférentiel produit une ontologie opérationnelle appropriée à la déduction : étant donné un ensemble initial de liens de parenté directs (père et mère), le SBC est capable d'inférer automatiquement toutes les relations familiales (parenté, fratrie, oncle, tante, nièce, neveu, etc.). Une autre opérationnalisation d'OntoFamily

dans un scénario de type validation produit un SBC capable d'évaluer la cohérence, complétude et concision d'une base d'assertions caractérisant l'arbre généalogique d'une famille particulière (cf. le site de TooCom pour télécharger les fichiers OCGL permettant de décliner les différentes ontologies opérationnelles à partir d'OntoFamily). Notons cependant que le processus d'opérationnalisation ne garantit pas la complétude, ni la cohérence de la base de connaissances obtenue.

2.4 MetaOCGL : une ontologie de représentation

Une *ontologie de représentation* définit les primitives d'un paradigme de représentation des connaissances (par exemple la Frame Ontology pour le paradigme des frames [16]). MetaOCGL est l'ontologie du langage OCGL, exprimée en OCGL. En ce sens, MetaOCGL peut être considérée comme une ontologie de représentation dédiée à la définition de la sémantique formelle d'un langage [15]. MetaOCGL inclut toutes les primitives d'OCGL et leurs relations (subsomption *isa*, exclusivité/incompatibilité entre relations, disjonction de concepts, liens entre les relations et les concepts d'un graphe exprimant un axiome), ainsi que les schémas d'axiomes et les axiomes de domaine qui expriment la sémantique formelle d'OCGL.

Comme le présente la figure 5, la hiérarchie de concepts de MetaOCGL inclut les primitives d'OCGL, *i.e.* `Concept`, `Relation` et `Property`. `Concept` et `Relation` sont spécialisés suivant un axe *Antécédent/Conséquent* permettant de différencier les primitives apparaissant en parties hypothèse et conclusion des axiomes de domaine. Ceci conduit à l'identification des concepts `Antecedent_C`, `Consequent_C`, `Antecedent_BR` et `Consequent _BR` pour les relations binaires, etc. Le concept `Property` regroupe tous les schémas d'axiome unaires d'OCGL (`Abstraction`, `Symmetry`, `Transitivity`, etc.). Les Schémas d'Axiomes binaires sont modélisés comme des relations dans la hiérarchie de relations de MetaOCGL. Cette hiérarchie contient donc les relations de subsomption (`ISA`, spécialisée en `ISA_C` et `ISA_R` pour la subsomption entre concepts et entre relations respectivement), les disjonctions de concepts, les exclusivités et incompatibilités entre relations et des relations dédiées à la signature de la primitive `Relation` d'OCGL (relation `role`). Les propriétés des concepts et relations de MetaOCGL sont spécifiées à la fois à travers des Schémas d'Axiomes décorant les hiérarchies (par exemple, la subsomption est irréflexive, antisymmétrique et transitive) et à travers des Axiomes de Domaine apparaissant en bas de la figure. Ces Axiomes de Domaine expriment essentiellement des propriétés d'héritage, comme par exemple l'héritage des propriétés algébriques par subsomption (`Algebraic Property Inheritance`). Il est important de noter que les Axiomes de Domaine de MetaOCGL contraignent fortement les représentations qu'il est possible de définir à l'aide d'OCGL (au même titre qu'une ontologie contraint les assertions possibles d'un domaine). Ainsi, dans le cadre d'OCGL, l'héritage des propriétés algébriques des relations est imposé.

Une ontologie de domaine peut être représentée par une instanciation de MetaOCGL (*i.e.* un graphe MetaOCGL), au même titre que des faits du domaine (*i.e.* des assertions) peuvent être représentés par des graphes OCGL. Un graphe MetaOCGL qui représente une ontologie de domaine contient un graphe dédié à la hiérarchie de concepts, un graphe dédié à la hiérarchie de relations et autant de graphes qu'il y a d'axiomes dans l'ontologie considérée. La figure 6 présente les graphes OCGL dédiés à la représentation de deux axiomes de OntoFamily *« Les*

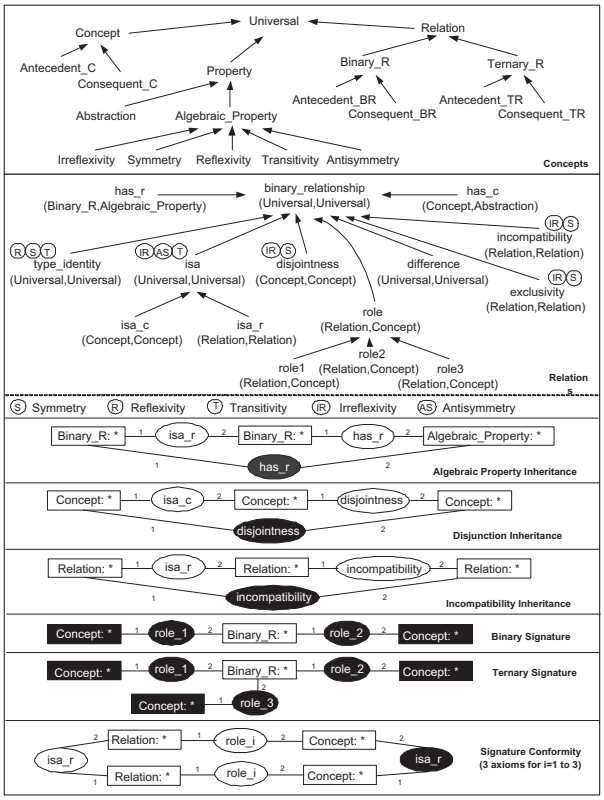

FIG. 5 – *Concepts, relations et axiomes de MetaOCGL. Afin de simplifier la présentation, nous avons volontairement limité le schéma aux relations ternaires. MetaOCGL permet cependant de prendre en compte des relations d'arité supérieure.*

ennemis de mes ennemis sont mes amis » et *« Les ennemis de mes amis sont mes ennemis »*, et leurs méta-graphes correspondant en MetaOCGL.

2.5 Ontologie de représentation, ontologie de domaine et SBC

Les ontologies de domaine étant des représentations d'un domaine, leur opérationnalisation produit des ontologies opérationnelles qui permettent de raisonner sur des assertions du domaine. De la même façon, raisonner sur les ontologies elles-mêmes peut être assuré par l'opérationnalisation de l'ontologie de représentation qui abstrait le formalisme utilisé. Plus précisément, si nous considérons des ontologies exprimées en OCGL par exemple, opérationnaliser l'ontologie d'OCGL (en l'occurrence MetaOCGL qui correspond à une ontologie dite de représentation) produit des ontologies opérationnelles permettant de raisonner sur les ontologies de domaine exprimées en OCGL. Opérationnaliser MetaOCGL consiste à choisir la façon dont les axiomes de MetaOCGL (*cf.* figure 5) vont être utilisés pour raisonner sur les ontologies exprimées en OCGL. Les versions opérationnelles de MetaOCGL peuvent ainsi

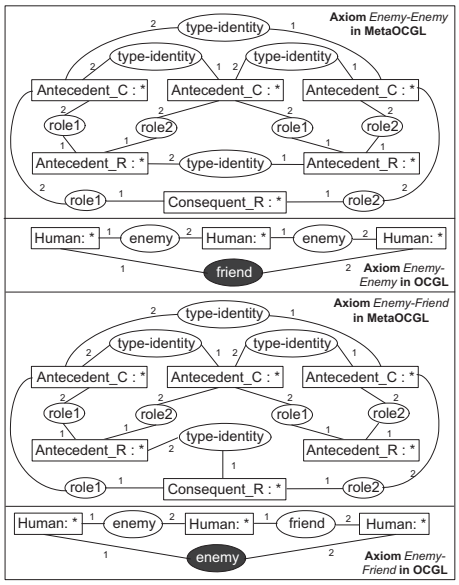

FIG. 6 – *Deux axiomes de OntoFamily représentés en MetaOCGL. Les liens d'identité de types (type_identity) au niveau méta dénotent le fait que les nœuds de l'axiome (au niveau domaine) sont de même type. Ainsi, sans considérer les liens d'identité de types, les deux méta-graphes sont identiques. Ils diffèrent en considérant ces liens d'identités car les relations de la partie hypothèse (appelées AntecedentR) de l'axiome du haut sont de même type et pas celles de l'axiome du bas. Ainsi, sans considérer les liens d'identité de types, les deux axiomes Enemy-Enemy et Enemy-Friend sont topologiquement identiques car leurs méta-graphes en MetaOCGL sont identiques.*

être utilisées pour faire de la complétion automatique dans une ontologie (ajout de liens de subsomptions, propagation de propriétés par héritage) dans le cas d'un scénario d'usage inférentiel, ou de la vérification d'ontologies par rapport à la sémantique formelle d'OCGL dans le cas d'un scénario de validation.

La figure 7 présente les interactions qui existent entre les ontologies de domaine, les ontologies de représentation et les Systèmes à Base de Connaissances. Elle souligne en particulier la place des trois principales activités relatives à l'intégration des ontologies au sein de SBC : *Modélisation*, *Opérationnalisation* et *Représentation*.

- *Activité de Modélisation*. Au niveau domaine, une ontologie (appelée Domain Ontology en figure 7) d'un domaine particulier (appelé Domain dans la figure) est définie via un processus de *Modélisation*. Raisonner sur des faits de ce domaine dans un SBC est rendu possible par l'opérationnalisation de l'ontologie selon un scénario d'usage particulier qui décrit la façon dont la partie axiomatique de l'ontologie va être utilisée dans le SBC. Pour résumer, la *Modélisation* d'un domaine conduit à une ontologie de domaine incluant *Concepts*, *Relations* et *Axiomes* (à la fois Schémas d'Axiome et Axiomes de Domaine) ;
- *Activité d'Opérationnalisation*. L'*Opérationnalisation* d'une ontologie de domaine construit le niveau ontologique d'un SBC, incluant les *connaissances terminologiques* (concepts

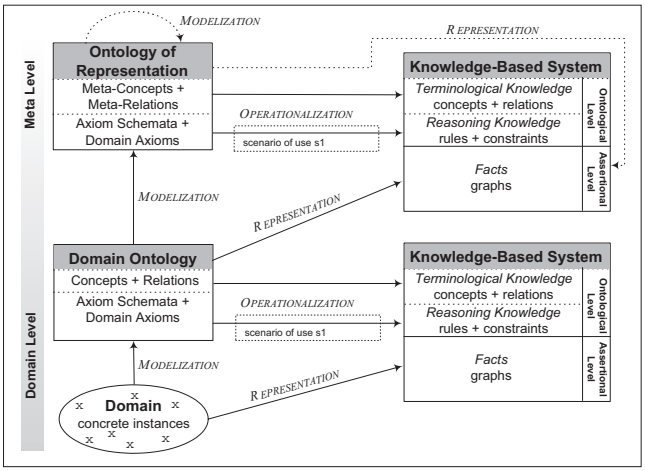

FIG. 7 – *Interactions entre ontologie de domaine, ontologie de représentation et SBC.*

et relations) et les *connaissances de raisonnement* (règles et contraintes) correspondant aux formes opérationnelles des axiomes dans le scénario d'usage choisi ;

– *Activité de Représentation*. Enfin, la *Représentation* du domaine conduit à la construction du *niveau assertionnel* du SBC, *i.e.* de la base de faits définie en accord avec les *connaissances terminologiques* et sur laquelle les *connaissances de raisonnement* vont pouvoir être mises en œuvre.

Ce processus en trois étapes, *Modélisation, Opérationnalisation, Représentation*, peut également être appliqué au niveau méta (*cf.* figure 7). L'ontologie de représentation modélise le langage utilisé pour exprimer les ontologies de domaine. Cette ontologie de représentation est aussi exprimée dans le langage considéré (*cf.* la boucle de modélisation au niveau méta). Elle peut être opérationnalisée dans un SBC, et l'ontologie opérationnelle générée permet de raisonner sur les ontologies de domaine. Dans ce SBC défini au niveau méta, un fait est une représentation d'une ontologie de domaine donnée, par exemple un graphe qui représente OntoFamily en MetaOCGL. Étant donné que l'ontologie de représentation est une méta-représentation, modéliser cette ontologie dans le même langage produit la même ontologie de représentation. Mais cette ontologie peut être représentée comme un fait dans un SBC qui en utilise une version opérationnelle, de façon à raisonner sur l'ontologie de représentation à l'aide d'elle-même. Cet aspect montre clairement que l'architecture que nous proposons (et les principes d'opérationnalisation sous-jacents) n'est pas limitée aux ontologies de domaine : elle peut s'appliquer quelle que soit la nature de l'ontologie considérée (ontologie de représentation, ontologie de PSM, etc.).

3 Aligner des ontologies denses

L'alignement d'ontologies de domaine est au cœur de la problématique de la gestion conjointe de plusieurs ontologies, devenue récemment nécessaire dans de nombreuses applications

requérant une interopérabilité sémantique, en particulier le Web sémantique. Les stratégies utilisées pour aligner deux ontologies sont diverses. À titre d'exemple, citons la classification hiérarchique [7], l'analyse formelle de concepts - Formal Concept Analysis - [21], l'analyse terminologique des noms de concepts et/ou des définitions en langage naturel [18] ou l'analyse structurelle des taxinomies de concepts [22]. Néanmoins, la plupart des travaux actuels portent exclusivement sur des ontologies qualifiées de légères (« *lightweight ontologies* »), *i.e.* des ontologies composées uniquement de taxinomies de concepts et de relations [15]. Aucun outil existant ne propose de fonctionnalités basées sur d'autres composants, en particulier les axiomes qui sont pourtant essentiels à la spécification de la sémantique des concepts et relations d'une ontologie [20].

L'objectif de l'alignement d'ontologies est de découvrir et d'évaluer des liens conceptuels (par exemple des identités, des subsomptions, ou des disjonctions) entre primitives conceptuelles (concepts et relations) de deux ontologies supposées bâties sur des domaines connexes [8]. Notre approche de ce problème repose sur l'utilisation du niveau axiomatique des ontologies pour découvrir des analogies sémantiques entre primitives, de façon à mettre en évidence des identités entre elles et à calculer les coefficients de vraisemblance de ces identités [11]. La comparaison des axiomes est basée sur leur représentation au niveau méta, de façon à préserver leur sémantique formelle tout en s'abstrayant de leurs différences terminologiques, alors que les algorithmes d'alignement existant sont principalement basés sur des comparaisons terminosyntaxiques [17].

Notre algorithme prend en entrée deux ontologies O_1 et O_2 (représentées en OCGL) et produit en sortie des identités potentielles entre 2 concepts ou 2 relations, identités pondérées par un coefficient de vraisemblance : le résultat est un ensemble d'appariements (P_i, P'_j, C), où P_i et P'_j sont respectivement des primitives conceptuelles de O_1 et O_2, et C le coefficient de vraisemblance de l'appariement[5]. Les Schémas d'Axiome comme les Axiomes de Domaine sont utilisés pour évaluer et/ou découvrir des appariements de primitives. Afin de pouvoir moduler les résultats en fonction du contexte (par exemple selon le type et le volume des ontologies considérées), nous avons attribué des poids par défaut pour chaque propriété d'OCGL. Ces poids peuvent être modifiés par l'utilisateur dans l'outil pour améliorer la pertinence du processus d'alignement. Par défaut, les valeurs des poids ont été ordonnées comme suit, à partir de considérations expérimentales : $W_{Sym} = W_{Trans} = W_{Refl} = W_{Irref} = W_{AntiSym} > W_{Disj} = W_{Incomp} = W_{Exclu} > W_{Cmin} = W_{Cmax} > W_{Axiom} > W_{Sign} > W_{Abst} > W_{ISA}$.

3.1 Utilisation des Schémas d'Axiome pour évaluer les appariements

Les Schémas d'Axiome qui impliquent seulement une primitive (*i.e.* propriétés algébriques et abstractions) sont comparés de O_1 à O_2, afin de découvrir des appariements de primitives. Si les primitives de l'appariement (p_1, p_2, c) portent toutes deux un même schéma d'axiome, le coefficient de vraisemblance c est augmenté du poids du schéma ; si l'appariement n'existe pas, il est créé avec ce poids ; si seule une des primitives porte le schéma, c est diminué du poids du schéma (ou l'appariement est créé avec le poids négatif s'il n'existe pas). Les schémas d'axiome qui impliquent deux primitives (*i.e.* disjonctions, incompatibilités et exclusivi-

[5] À l'heure actuelle, seule la relation d'identité est prise en compte dans nos travaux.

tés) sont utilisés de la même façon, mais en considérant les 4 appariements possibles des 2 primitives.

	Relation r_1 in Ontology 1	Relation r_2 in Ontology 2	Action
Min Card	0 (resp. $c_{min} \geqslant 1$)	$c_{min} \geqslant 1$ (resp. 0)	$-2 * W_{cmin}$
	$c_{min} \neq 0$	$c_{min} \neq 0$	$+2 * W_{cmin}$
	$c1_{min} \neq 0$	$c2_{min} \neq 0$ and $\neq c1_{min}$	$-W_{cmin}$
	Relation r_1 in Ontology 1	Relation r_2 in Ontology 2	Action
Max Card	∞ (resp. $c_{max} \geqslant 1$)	$c_{max} \geqslant 1$ (resp. ∞)	$-W_{cmax}$
	$c_{max} \neq \infty$	$c_{max} \neq \infty$	$+2 * W_{cmax}$
	$c1_{max} \neq \infty$	$c2_{max} \neq \infty$ and $\neq c1_{max}$	$-2 * W_{cmax}$

TAB. 1 – *Modifications du coefficient de l'appariement* (r_1, r_2, c) *en fonction des cardinalités des relations (*c_{min} *et* c_{max} *sont les valeurs des cardinalités).*

Enfin, le tableau 1 présente les différentes opérations induites par la prise en compte des égalités ou différences de cardinalités. Si l'appariement entre les relations considérées n'existe pas alors qu'une analogie entre cardinalités est découverte, l'appariement est créé avec le coefficient correspondant.

3.2 Utilisation des Axiomes de Domaine pour évaluer les appariements

Comme indiqué en figure 6, les axiomes de domaine sont représentés en MetaOCGL, afin de comparer leurs structures, indépendamment des labels portés par les nœuds. Pour chaque couple d'axiomes (a_1, a_2), où $a_1 \in O_1$ et $a_2 \in O_2$, les représentations de a_1 et a_2 en MetaOCGL, $meta(a_1)$ et $meta(a_2)$, sont construites. Ces représentations peuvent être enrichies par l'ajout d'informations sur les nœuds : par exemple, dans la figure 6, les deux relations *enemy* de l'axiome *Enemy-Enemy* en OCGL sont représentées en MetaOCGL par les deux concepts *Antecedent_R* qui sont liés par la méta-relation *type_identity*.

Deux types d'équivalence topologique entre axiomes sont considérés :

- l'**équivalence**, qui est attestée lorsque deux projections (au sens de l'homomorphisme de graphes du modèle des GCs) existent de $meta(a_1)$ dans $meta(a_2)$ et de $meta(a_2)$ dans $meta(a_1)$, sans considérer les relations $type\_identity$;
- l'**équivalence typée**, qui est attestée lorsque les deux projections existent en considérant les relations $type\_identity$.

L'existence d'une équivalence (resp. équivalence typée) accroit du poids de l'équivalence (resp. équivalence typée) le coefficient des appariements entre primitives liées par les projections. Bien entendu, le poids d'une équivalence typée est plus important que celui d'une équivalence simple. Par exemple, les deux axiomes de la figure 6 sont équivalents car 2 projections existent entre leurs méta-graphes sans considérer les relations *type-identity*. En considérant les relations *type-identity*, il n'existe pas de projection, les axiomes ne sont donc pas équivalents typés.

3.3 Résolution des conflits d'appariements

Du fait qu'elles lient une relation et un ensemble de concepts, les signatures sont uniquement utilisées pour augmenter ou diminuer les coefficients des appariements, et non pour en créer de nouveaux. En outre, deux usages possibles sont distingués : (1) la modification des coefficients d'appariements de relations à partir des appariements entre concepts de leurs

signatures et (2) la modification des coefficients d'appariements de concepts à partir des appariements entre relations possédant ces concepts dans leurs signatures. Par exemple, si les relations *mother1(Woman1, Human1)* de O_1 et *parent2(Human2, Human2)* de O_2 sont appariées, les coefficients des appariements de concepts *(Woman1, Human2)* et *(Human1, Human2)*, s'ils existent, sont augmentés de W_{Sign}. Mais si les appariements *(Woman1, Human2)* et*(Human1, Human2)* existent, le coefficient de l'appariement de relations *(mother1, parent2)* peut aussi être augmenté de W_{Sign}. Le choix entre ces deux possibilités est basé sur l'utilisation d'un **Matching Ratio** appelé MR_c (resp. MR_r) et défini entre le nombre d'appariements de concepts (resp. relations) et le nombre moyen de concepts (resp. relations) dans O_1 et O_2 : si MR_c est supérieur à MR_r (*i.e.* la proportion d'appariements de concepts découverts est supérieure à celle des appariements de relations), les signatures sont utilisées pour réévaluer les coefficients des appariements de relations existant à l'aide des appariements de concepts existant, et vice versa. Ainsi, avant d'utiliser les signatures, les appariements de concepts (ou de relations) doivent être résolus, et ensuite les signatures sont utilisées pour affiner les coefficients des appariements de relations (ou de concepts).

La résolution des appariements de concepts ou de relations repose sur les valeurs de leurs coefficients : lorsque plusieurs appariements existent pour une primitive donnée, l'appariement avec le coefficient le plus élevé est retenu. Bien entendu, des contradictions peuvent exister entre 2 ou plusieurs appariements, que les valeurs des coefficients ne permettent pas toujours de résoudre. Dans ce cas, l'intervention de l'utilisateur est nécessaire pour trancher.

3.4 Résultats expérimentaux

Nous ne présentons ici ces premiers résultats que dans les grandes lignes, une présentation détaillée est disponible dans [**?**]. L'application de notre algorithme sur deux ontologies du domaine des relations familiales (construites en parallèle par deux groupes d'étudiants) OntoFamily O_1 et OntoFamily O_2 produit 201 appariements, dont 9 couples de concepts (parmi 9 possibles), et 192 couples de relations (parmi 713 possibles)[6]. Beaucoup d'appariements n'apparaissent qu'à l'occasion de moins de 4 comparaisons d'Axiomes de Domaine, et jamais lors de la comparaison de Schémas d'Axiomes. Ils sont donc rejetés du fait de leur faible pertinence et seuls 9 couples de concepts et 69 couples de relations sont ainsi considérés pour la phase de résolution.

Les valeurs des rapports MR_c et MR_r sont respectivement de 3 et ≈ 2.55, les couples de concepts sont donc résolus avant les couples de relations. Le couple *(Human,Human)* possède le coefficient le plus élevé des appariements liant les concepts *Human*. Ensuite, le couple de plus fort coefficient est *(Woman,Female)* et le troisième couple de plus fort coefficient est *(Man,Male)*. Tous les liens entre concepts des 2 ontologies sont donc retrouvés par l'algorithme. Les signatures sont ensuite utilisées pour affiner les valeurs des coefficients des couples de relations. La résolution des couples de relations ainsi constitués conduit à 9 liens entre relations de O_1 et O_2 (parmi 17 liens réellement possibles) : *(aunt,aunt), (daughter, daughter), (enemy,enemy), (father,father), (friend,friend), (husband,husband), (son,son), (uncle,uncle)* et *(wife,wife)*. La figure 8 présente les résultats de cette expérience au sein de l'atelier TooCom.

[6] À titre indicatif, OntoFamily O_1 est composé de 3 concepts, 31 relations binaires (structurées en treillis de profondeur 3), 11 Schéma d'Axiomes et 18 Axiomes de Domaine. OntoFamily O_2 est composé de 3 concepts, 23 relations binaires (structurées en arbre de profondeur 2), 10 Schéma d'Axiomes et 27 Axiomes de Domaine.

Il est important de noter que les poids associés à chaque propriété d'OCGL peuvent être directement modifiés via l'interface qui adapte en conséquence et simultanément les résultats.

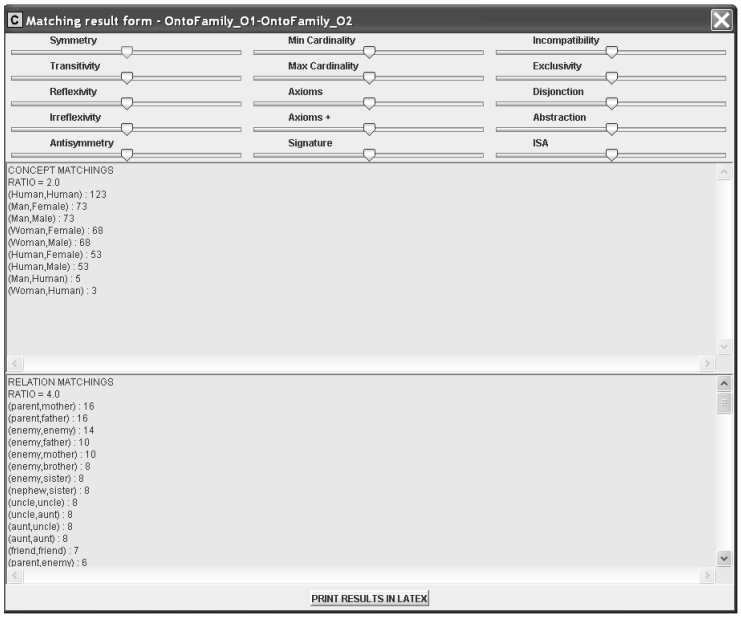

FIG. 8 – *Extrait de résultats du processus d'alignement dans TooCom.*

Ces résultats, bien que perfectibles, s'avèrent encourageants étant donné le faible niveau de complexité des 2 ontologies considérées. En effet, notre approche pourrait s'avérer encore plus pertinente dans un contexte d'ontologies plus hétérogènes et plus riches à la fois en termes de concepts, relations et axiomes. Il est important de noter qu'il existe plusieurs benchmarks dédiés à la comparaison des méthodes d'alignement, les plus (re)connus étant ceux proposés lors des campagnes OAEI "Ontology Alignment Evaluation Initiative" [3] (http ://oaei.ontology-matching.org/). Le problème est que les ontologies fournies dans ces campagnes d'évaluation sont uniquement des ontologies légères. Par exemple, pour l'édition 2006, l'ontologie en anatomie incluait certes plus de 10.000 concepts et relations, mais seulement 2 propriétés algébriques sur les relations (en l'occurrence des symétries), et bien évidemment aucun axiome de domaine. Dans ces conditions, il n'est pas pertinent d'appliquer notre méthode. Nous travaillons actuellement à la définition d'une ontologie dense dans le domaine HSE (Hygiène, Sécurité et Environnement).

4 Conclusion

Dans ce papier, nous avons présenté un atelier dédié à la représentation, l'opérationnalisation et l'alignement d'ontologies denses (*i.e.* des ontologies intégrant les règles et les contraintes régissant les domaines considérés et décrits à l'aide de hiérarchies de concepts et de relations). Cet atelier se distingue des autres ateliers d'ingénierie des ontologies tels

que Protégé[7] ou WebODE[8] de part la prise en compte explicite et au niveau connaissance des axiomes du domaine considéré. Ainsi, dans Protégé, les axiomes ne sont pas représentés et pris en compte au niveau connaissance, mais directement représentés sous une forme opérationnelle à l'aide du formalisme logique PAL (Protégé Axioms Language). Ils sont ainsi difficilement appréhendables par l'ingénieur de la connaissance (qui n'est pas nécessairement un logicien), et par conséquent difficile à évaluer et mettre à jour. La représentation graphique sous-jacente à TooCom facilite la visualisation et la lisibilité de la base d'axiomes. Par ailleurs, la mise en oeuvre de raisonnements à l'aide d'opérations de graphes facilite la compréhension des interactions entre les axiomes. Ces deux aspects contribuent au développement d'un atelier d'ingénierie des ontologies denses facilitant le prototypage au niveau connaissance, *i.e.* construction itérative et interactive de la base d'axiomes (règles et contraintes du domaine).

Nos travaux se poursuivent actuellement sur la définition formelle, dans le cadre du modèle des GC, de différentes notions qui s'avèrent indispensables à la mise en oeuvre de plusieurs activités relevant de l'ingénierie des ontologies denses, en particulier la vérification (minimalité, complétude et cohérence d'une base d'axiomes) et la fusion (identification d'un ensemble cohérent, complet et non-redondant d'axiomes, sous-ensemble des deux bases d'axiomes issus des deux ontologies). Quatre notions ont déjà été formalisées [13] : (i) *Axiomes Partiellement Compatibles*, (ii) *Axiomes Totalement Incompatibles*, (iii) *Axiomes Incompatibles* et (iv) *Spécialisation/Généralisation d'Axiomes*. Elles sont en cours d'implémentation au sein de l'atelier TooCom.

Références

[1] J.F. Baget and M.L. Mugnier. Extensions of simple conceptual graphs : The complexity of rules and constraints. *Journal of Artificial Intelligence Research (JAIR)*, 16 :425–465, 2002.

[2] S. Bechhofer, R. Volz, and P. Lord. Cooking the Semantic Web with the OWL API. In *Proceedings of the second International Semantic Web Conference (ISWC'2003)*, 2003.

[3] R. Benjamins, J. Euzenat, N. Noy, P. Shvaiko, H. Stuckenschmidt, and M. Uschold. Ontology alignment evaluation initiative - 2006. In *Proceedings of the International Workshop on Ontology Matching*, http ://oaei.ontologymatching.org/2006/, 2006.

[4] T. Berners-Lee, J. Handler, and O. Lassila. The semantic web. In *Scientific American*, volume 248, pages 35–43, 2001.

[5] B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins. The Ontology of Tasks and Methods. In *Proceedings of the 11th workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, 1998.

[6] M. Chein and M.L. Mugnier. Conceptuals graphs : fundamental notions. *Revue D'Intelligence Artificielle (RIA)*, 6(4) :365–406, 1992.

[7] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology Matching : A Machine Learning Approach. In *Handbook on Ontologies in Information Systems*, pages 397–416, 2004.

[7] http ://protege.stanford.edu
[8] http ://webode.dia.fi.upm.es

[8] J. Euzenat. An API for ontology alignment. In *Proceedings of the 3rd International Semantic Web Conference (ISWC'2004)*, pages 698–712. Springer Verlag LNCS 3298, 2004.

[9] F. Fürst. TooCoM : a Tool to Operationalize an Ontology with the Conceptual Graph Model. In *Proceedings of the Workshop on Evaluation of Ontology-Based Tools (EON'2003) at ISWC'2003*, pages 57–70, 2003.

[10] F. Fürst, M. Leclère, and F. Trichet. Operationalizing domain ontologies : a method and a tool. In R. Lopez de Mantaras and L. Saitta, editors, *European Conference on Artificial Intelligence (ECAI'2004)*, pages 318–322. IOS Press, 2004.

[11] F. Fürst and F. Trichet. Axiom-based ontology matching. In *3rd International Conference on Knowledge Capture (K-CAP'2005, Banff-Canada)*. ACM Press. ISBN 1-58113-380-4, 2005.

[12] F. Fürst and F. Trichet. Toocom : bridge the gap between ontologies and knowledge-based systems. In *17th International Conference on Knowledge Engineering and Software Engineering (SEKE'2005)*, pages 235–243. KSI editors, 2005.

[13] F. Fürst and F. Trichet. Heavyweight ontology engineering. In *5th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE'2006). Lecture Notes in Computer Science - LNCS 4277*, pages 38–39. Springer-Verlag, 2006.

[14] D. Genest and E. Salvat. A platform allowing typed nested graphs : how CoGITo became CoGITaNT. In *Proceedings of the 6th International Conference on Conceptual Structures*, pages 154–161, http ://cogitant.sourceforge.net, 1998. Springer-Verlag (LNAI 1453).

[15] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering*. Springer, Advanced Information and Knowledge Processing, 2003.

[16] T.R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2) :199–220, 1993.

[17] Y. Kalfoglou and M. Schorlemmer. Ontology mapping : the state of the art. *The Knowledge Engineering Review*, 18(1) :1–31, 2003.

[18] N. F. Noy and M. Musen. The PROMPT suite : Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6) :983–1024, 2003.

[19] J. Sowa. *Conceptual Structures : information processing in mind and machine*. Addison-Wesley, 1984.

[20] S. Staab and A. Maedche. Axioms are objects too : Ontology engineering beyong the modeling of concepts and relations. Research report 399, Institute AIFB, Karlsruhe, 2000.

[21] G. Stumme and A. Maedche. FCA-MERGE : Bottom-up merging of ontologies. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'2001)*, pages 225–234, 2001.

[22] B. Thanh Le, R. Dieng-Kuntz, and F. Gandon. On Ontology Matching Problems - for Building a Corporate Semantic Web in a Multi-Communities Organization. In *Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS'2004)*, volume 4, pages 236–243, 2004.

Annexe 1. Algorithme d'alignement basé sur les axiomes

INPUT. Two ontologies O_1 and O_2 expressed in OCGL, $O_1 = \{C_1, R_1, SA_1, DA_1\}$ and $O_2 = \{C_2, R_2, SA_2, DA_2\}$ where C_i is a set of concepts, R_i is a set of relations, SA_i is a set of schemata axioms about the concepts of C_i and the relations of R_i and DA_i is a set of domain axioms about the concepts of C_i and the relations of R_i ; O_1 is supposed to be the reference ontology (we assume that the end-user has favoured O_1 for the matching process)

OUTPUT. A set of primitive matchings $M = \{(p_i, p'_i, w)_{i=1..m} \mid (p_i, p'_i) \in C_1 \times C_2$ or $(p_i, p'_i) \in R_1 \times R_2$ and w is the weight of the matching$\}$

{First Step : discovering and valuating matchings by using Schemata Axioms}
for each schemata axiom $SA \in$ {algebraic properties of relations, abstraction of concepts, disjointness of concepts, exclusivity and incompatibility between relations, cardinalities of relations} with weight W_{SA} **do**
  for each couple (p_1,p_2) of primitives of $C_1 \times C_2$ or $R_1 \times R_2$ **do**
    if SA is applied to p_1 {p_1 bears SA} **then**
      if SA is applied to p_2 {p_1 bears SA} **then**
        if the matching (p_1,p_2,w) exists in M **then**
          $w = w + W_{SA}$
        else
          the matching (w_1,w_2,W_{SA}) is created and added to M
        end if
      else
        if the matching (p_1,p_2,w) exists in M **then**
          $w = w - W_{SA}$
        else
          the matching (p_1,p_2,$-W_{SA}$) is created and added to M
        end if
      end if
    end if
  end for
end for

{Second Step : valuating matchings by using Domain Axioms}
for each domain axiom a_1 of DA_1 **do**
  for each domain axiom a_2 of DA_2 **do**
    $m_1 = meta(a_1)$ {the meta-representation of a_1 with MetaOCGL ontology}
    $m_2 = meta(a_2)$ {the meta-representation of a_2 with MetaOCGL ontology}
    if there exists at least a simple projection between m_1 and m_2 **and** there exists at least a simple projection between m_2 and m_1 **then** {a_1 and a_2 are simply equivalent}
      for each existing matching (p_1, p_2,w) in M, of primitives linked by the projections **do**
        $w = w + W_{topo}$ {W_{topo} is the weight modifier of simply equivalence}
      end for
    end if
    if there exists at least a typed projection between m_1 and m_2 **and** there exists at least a typed projection between m_2 and m_1 **then** {a_1 and a_2 are typed-equivalent}
      for each existing matching (p_1, p_2,w) in M, of primitives linked by the projections **do**
        $w = w + W_{topo+}$ {W_{topo+} is the weight modifier of typed-equivalence}
      end for

```
      end if
    end for
  end for

  {Third Step : fixing concept or relation matchings}
  MR_r = (number of relation matchings in M)/((|R_1| + |R_2|)/2)
  MR_c = (number of concept matchings in M)/((|C_1| + |C_2|)/2)
  if MR_r ≤ MR_c then
    for each concept c of C_1 do
      get all existing matchings (c, c'_i, w_i)_{i=1..n} in M
      only keep in M the most valuated matching (eventually with user help)
    end for
  else
    for each relation r of R_1 do
      get all existing matchings (r, r'_i, w_i)_{i=1..m} in M
      only keep in M the most valuated matching (eventually with user help)
    end for
  end if

  {Fourth Step : valuating matchings by using relation signatures}
  for each existing matching (r(c_1, .., c_n), r'(c'_1, .., c'_n), w) in M where r ∈ R_1 and r' ∈ R_2 do
    if MR_r ≤ MR_c then {the signatures are used to improve relation matchings by using concept
    matchings}
      if all the matchings (c_i, c'_i, w_i)_{i=1..n} exist in M then
        w = w + W_sign {W_sign is the weight modifier of signature}
      else
        w = w − W_sign
      end if
    else {the signatures are used to improve concept matchings by using relation matchings}
      for each i = 1..n do
        if the matching (c_i, c'_i, w_i) exists in M then
          w_i = w_i + W_sign
        end if
      end for
    end if
  end for

  {Fifth Step : fixing remaining matchings}
  if MR_r ≥ MR_c then
    for each concept c of C_1 do
      get all existing matchings (c, c'_i, w_i)_{i=1..n} in M
      only keep in M the most valuated matching (eventually with user help)
    end for
  else
    for each relation r of R_1 do
      get all existing matchings (r, r'_i, w_i)_{i=1..n} in M
      only keep in M the most valuated matching (eventually with user help)
    end for
  end if
```

Summary

An *heavyweight ontology* is a *lightweight ontology* (*i.e.* an ontology simply based on a hierarchy of concepts and a hierarchy of relations) enriched with axioms used to fix the semantic interpretation of concepts and relations. Such an ontology can be a domain ontology, an ontology of representation, an ontology of PSM, etc. In our work, we argue in favor of using a graph-based solution to deal with the different activities related to Heavyweight Ontology Engineering, in particular ontology representation, ontology operationalisation, ontology evaluation (*i.e.* verification and validation) and ontology matching. Our approach consists in using the graph-based paradigm to represent all the components of an heavyweight ontology (*i.e.* Concepts, Relations and Axioms) and using graph homomorphism techniques to compare (at the conceptual level) the core components of an heavyweight ontology: the Axioms. This explicit graph-based representation of axioms coupled with reasoning capabilities based on graphs homomorphism facilitates both (1) the definition of important notions for Heavyweight Ontology Engineering such as *Compatible/Incompatible Axioms* or *Specialisation/Generalisation of Axioms* and (2) the topological comparison of axioms, which in our work is used to define a new approach of ontology matching mainly based on axiom-based ontology morphisms. All these features are implemented in the TooCom framework (*Tool to Operationalize an Ontology with the Conceptual Graphs Model*) available on the Web at *http://sourceforge. net/projects/toocom/*.