Introduction de raisonnement dans un outil de gestion de connaissances basé sur les Topic Maps

Olivier Carloni, Michel Leclère, Marie-Laure Mugnier

Introduction de raisonnement dans un outil de gestion des connaissances basé sur les Topic Maps

Olivier Carloni[1,2], Michel Leclère[1], Marie-Laure Mugnier[1]

[1] LIRMM, CNRS - Université Montpellier 2
161 rue Ada, F-34392 Montpellier cedex 5 - France
{carloni, leclere, mugnier}@lirmm.fr

[2] Mondeca,
3 cité Nollez, F-75018 Paris - France
http://www.mondeca.com

Résumé Cet article présente les premières étapes de conception d'un moteur d'inférence pour un outil industriel de gestion des connaissances. Le logiciel existant permet la construction et l'exploitation de bases de connaissances comportant un méta-modèle, des ontologies et des instances en s'appuyant sur le langage des Topic Maps. Les Topic Maps ne disposant pas de sémantique formelle mais reposant sur un modèle de graphes étiquetés, nous proposons de nous appuyer sur le formalisme des graphes conceptuels pour la mise en œuvre de raisonnement sur ces bases. Nous présentons une transformation réversible des Topic Maps vers les graphes conceptuels qui nous permet à la fois de doter le langage des Topic Maps d'une sémantique formelle, d'exploiter les mécanismes de raisonnement des graphes conceptuels et de réutiliser l'API CoGITaNT pour l'implémentation des mécanismes d'inférences.

Mots-clés : Raisonnement, Topic Maps, Graphes Conceptuels

1 Introduction

Depuis une dizaine d'années, la connaissance a fait une entrée massive dans les organisations industrielles et administratives, répondant aux besoins (1) de capitalisation du patrimoine immatériel des organisations que constituent les savoirs et savoirs-faire de leurs membres et (2) d'exploitation à un niveau "sémantique" des systèmes d'information des organisations c'est-à-dire une exploitation s'appuyant sur la sémantique des informations échangées. Cet engouement a ouvert la voie à une véritable ingénierie des connaissances dont l'idée principale est d'éliciter et représenter les connaissances dans un langage formel et de fixer l'interprétation des symboles du langage à l'aide d'ontologies (ontologies de domaine pour le vocabulaire conceptuel et ontologies de représentation pour les constructeurs du langage) agissant comme des référentiels sémantiques.

Mondeca est un éditeur de logiciels qui développe ITM (Intelligent Topic Manager) un logiciel de gestion des connaissances fondé sur ce principe. Le cœur de ce logiciel est une base de connaissances structurée en 3 niveaux :

1. Le niveau méta contient l'ontologie de représentation. Il comporte toutes les primitives permettant de créer des ontologies de domaine pour chaque client de l'outil ITM. Ce niveau évolue très peu et est commun à tous les clients.
2. Au niveau modèle, des ontologies sont créées pour chaque client en utilisant les primitives de l'ontologie de représentation. On définit ici des types et des propriétés générales qui seront utilisées au niveau des instances. Plusieurs sortes d'ontologies sont généralement définies : une ontologie de domaine spécifique au client, des ontologies spécifiques "de services" (primitives de spécification de thésaurus, primitives de description d'une organisation logique des documents...) permettant la mise en oeuvre des services offerts par ITM.
3. Le niveau des instances contient des annotations sémantiques, thésaurus, et des descriptions organisationnelles construites en utilisant le vocabulaire conceptuel des ontologies du niveau modèle. Ce niveau sera géré par le client.

ITM s'appuie sur sa base de connaissances pour fournir de nombreux services de gestion, d'indexation, d'annotation, de navigation...

Le Lirmm et Mondeca collaborent à un projet de mise en œuvre de raisonnements sur les connaissances des bases ITM afin de permettre une automatisation des vérifications des connaissances ajoutées par l'utilisateur au niveau instance et d'étendre les capacités d'interrogation de la base de connaissances. Cet objectif nécessite de fixer la sémantique formelle des représentations utilisées dans ITM afin de définir les raisonnements visés. Le langage de représentation utilisé dans ITM est basé sur les "Topic Maps" (TM) [9]. Les TM peuvent être vues comme des graphes (ou des

hypergraphes) étiquetés [2] dont les sommets sont les topics et associations, et les arêtes indiquent la participation d'un topic à une association (cf. section 2). Le langage des TM ne dispose pas de sémantique formelle propre, cependant les proximités qu'il entretient avec les formalismes de graphes conceptuels [11] et en particulier la famille $\mathcal{SG}$ [3] – une TM pouvant être naturellement traduit en un graphe conceptuel simple (SG) – nous ont amenés à plonger les représentations d'ITM dans cette famille de langages.

Cette assimilation des TM à des SG permet de doter ITM d'une sémantique formelle puisque la famille $\mathcal{SG}$ est fondée par rapport à la logique des prédicats, de prendre en compte des règles d'inférence dans les bases ITM car la famille $\mathcal{SG}$ dispose de différents types de connaissances (règles, contraintes, définitions...), de récupérer les mécanismes de raisonnement de la famille $\mathcal{SG}$ qui sont basés sur des opérations de graphes (facilitant ainsi leur interprétation) tout en étant adéquats et complets vis à vis de la déduction en logique des prédicats, de bénéficier des algorithmes efficaces développés pour ces opérations de graphes et d'utiliser la bibliothèque GPL CoGITaNT de développement d'applications à base de graphes conceptuels [6]. Notre démarche peut être rapprochée de celle mise en oeuvre dans l'outil Corese [7], qui traite des données RDF/S par un moteur graphes conceptuels.

Dans cet article, nous présentons la transformation des TM d'ITM dans la famille $\mathcal{SG}$ et le langage de requêtes sur le réseau de topics – ainsi que les mécanismes de calcul des réponses à ces requêtes – que cette transformation nous a permis d'obtenir dotant ainsi ITM d'un premier module de raisonnement qualifié formellement. En section 2, le langage des TM, l'architecture à trois niveaux des bases ITM sont présentés ; cette section se termine par une brève présentation des services d'ITM et des scénarios de mise en œuvre de raisonnnement envisagés. La section 3 est dédiée à la présentation de la famille $\mathcal{SG}$. En section 4, la transformation, les raisonnements qu'elle permet de mettre en œuvre et les apports pour ITM sont détaillés ; enfin la section 5 détaille l'architecture d'intégration du moteur d'inférence à ITM, présente les premiers résultats expérimentaux et trace quelques perspectives d'extension du travail réalisé.

2 L'outil ITM de gestion des connaissances

ITM intègre des connaissances de diverses natures (ontologie, thésaurus, taxinomie) afin de répondre aux différents besoins des entreprises dans le domaine de la gestion des connaissances. Sa solution s'appuie sur une base de connaissances structurée en 3 niveaux sur laquelle ont été construits différents services : indexation terminologique des documents à partir du thesaurus, gestion de la structure organisationnelle des documents, construction d'annotations contrôlée par l'ontologie de domaine, annotation semi-automatique de documents, navigation dans la base d'annotations, recherche dans la base d'annotations (cf. figure 1).

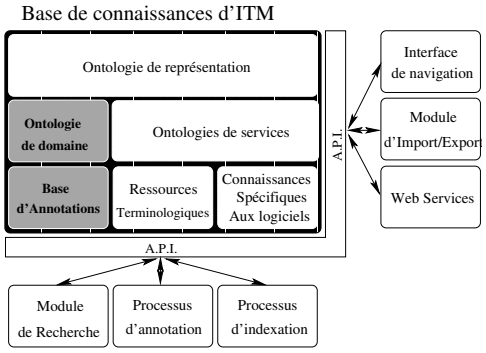

FIG. 1 – La base et les modules d'ITM.

ITM utilise le langage des TM pour représenter et stocker les connaissances de ses bases. La connaissance est décrite par un ensemble de *topics* représentant des entités du domaine modélisé. Ces topics sont reliés entre eux par des *associations* qui identifient le *rôle* joué par chaque topic. Les topics peuvent être identifiés par des *noms* et caractérisés par des *occurrences* qui sont utilisées dans ITM comme des couples attribut-valeur décrivant le topic. Deux associations spécifiques sont normalisées pour toute TM : l'association *classe-instance* et l'association *superclasse-sousclasse*.

2.1 Le standard TM

Les TM sont un langage standardisé ayant pour objectif de permettre la structuration d'un ensemble de ressources (documents addressables) à l'aide de topics qui rassemblent des ressources partageant des propriétés communes et d'associations indiquant des liens entre topics [9]. Ce langage dispose d'une syntaxe XML [12] qui en fait un des langages candidats à une utilisation dans le cadre du web sémantique.

Contrairement à ces concurrents directs, RDF, RDFS et la famille OWL, les TM n'ont pas été munies d'une semantique formelle et, à notre connaissance, aucun outil de raisonnement n'a été proposé pour ce langage.

Une TM est composée d'un ensemble de topics reliés entre-eux par des associations. Les associations sont typées et identifient le rôle joué par chaque topic dans l'association. Afin de faciliter la lecture des TM dans ce papier, nous avons choisi de représenter l'association particulière *classe-instance* comme une étiquette de topic. Une TM est donc un graphe biparti étiqueté $tm = (T, A, E, type, nom)$ composé de sommets topics T et de sommets associations A. Les arêtes $E \subseteq A \times T$ indiquent la participation des topics aux associations. $type$ et nom sont des fonctions d'étiquetage de $T \cup A \cup E$ dans L_T où L_T est un ensemble de libellés identifiant des topics. La fonction $type$ associe à chaque sommet et arête un libellé de L_T spécifiant selon le

cas la classe (d'un topic), le type (d'une association) et le rôle (d'un lien de participation). La fonction partielle nom permet d'identifier certains topics par un nom.

La figure 2 présente une TM décrivant un topic de classe `Société` et de nom `Oracle` qui joue le rôle `acquéreur` dans une association de type `acquisition` la reliant à deux autres topics.

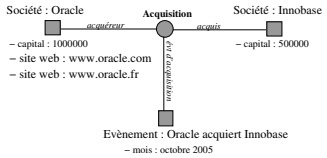

FIG. 2 – Une TM.

On peut compléter la description d'un topic par un ensemble d'occurrences qui indiquent une information pertinente pour le topic. Chaque occurrence est composée d'une valeur v, d'un type de données d (appelé son *type physique* dans ITM), et du nom l du topic qui type l'occurrence (appelé *type logique* dans ITM). On dispose donc d'une fonction occ de T dans $2^{V \times D \times L_T}$ qui associe à un topic l'ensemble de ses occurrences, où D est l'ensemble des types de données, et V l'ensemble des valeurs des types de D.

La TM de la figure 2 associe au topic `Oracle` trois occurrences : la valeur de type `entier` comme `capital`, et les valeurs `www.oracle.com` et `www.oracle.fr` de type `URL` comme `site web`. Ces occurrences peuvent s'interpréter comme le capital de la société Oracle et les adresses web permettant de la contacter.

Notations. Soit Ref la fonction qui associe à une TM le sous-ensemble des libellés de L_T auxquels font référence ses fonctions $type$ et occ, et soit Nom, l'extension de la fonction nom à une TM, qui associe à une TM le sous-ensemble des libellés de L_T images d'un topic par nom. Une TM tm est dite *complète* si tout libellé utilisé par les fonctions $type$ et occ est associé à un topic par la fonction nom : $Ref(tm) \subseteq Nom(tm)$.

2.2 Les bases ITM

Une base de connaissances ITM forme une TM complète structurée en trois niveaux par l'association *classe-instance* :

- le niveau méta décrit le méta-modèle réflexif des connaissances d'ITM (cf. cadre `Niveau Méta` de la figure 3) ;
- le niveau modèle constitué des différents modèles utilisés dans une base ITM qui sont décrits en utilisant les primitives du niveau méta. Ce niveau est généralement composé d'une ontologie de domaine spécifiant la sémantique du vocabulaire utilisé pour décrire (au niveau sémantique) le contenu des informations gérées par la base, et de plusieurs ontologies de "services" spécifiant la sémantique des représentations dédiées à la mise en œuvre de services spécifiques d'ITM (primitives de représentation de thesaurus, primitives de représentation de l'organisation logique des documents...) ;
- enfin le niveau instances contient les instances de la base qui sont "contrôlées" par un modèle du niveau supérieur. On y trouve des annotations sémantiques qui décrivent le contenu des informations gérées par la base, des ressources terminologiques (thesaurus), la description de l'organisation logique des documents...

Les niveaux modèle et instance sont par ailleurs découpés en "espaces de travail" dédiés à un type de connaissance : annotation sémantique, thésaurus, structure organisationnelle des documents... Pour chaque espace e, on trouve une TM au niveau modèle, tm_e^{modele}, qui décrit le vocabulaire utilisable au niveau instances pour représenter les connaissances spécifiques de l'espace, et une TM associée, $tm_e^{instance}$, au niveau instances qui représente ces connaissances. Ces différentes TM respectent les contraintes suivantes :

- tm^{meta} est elle-même une TM complète ;
- tous les topics des TM des niveaux méta et modèle sont identifiés par un nom (i.e. la fonction nom est totale pour ces TM) ;
- les références utilisées dans les TM respectent la hiérarchie des niveaux et restent internes[1] aux espaces de travail, *i.e. pour chaque espace* e, $Ref(tm_e^{modele}) \subseteq Nom(tm^{meta})$ et $Ref(tm_e^{instance}) \subseteq Nom(tm_e^{modele})$.

La figure 3 présente un exemple des 3 niveaux d'une base ITM dédiée à l'intelligence économique. Les niveaux modèle et instance sont des extraits des TM de l'espace dédié aux annotations sémantiques : on y trouve donc un extrait de l'ontologie et du réseau de connaissances sur les entreprises. Le niveau supérieur générique à toute base ITM exprime de manière réflexive le méta-modèle ITM. Les connaissances de cette tm^{meta} sont interprétées pour doter les bases de connaissances ITM de la sémantique opérationnelle suivante :

- une association de type `type d'association permis` reliant trois topics t_c, t_a et t_r spécifie qu'il est permis au niveau inférieur à un topic de classe t_c de jouer le rôle t_r dans l'association de type t_a. Par exemple, le niveau modèle de la figure 3 autorise un topic de classe `Société` à jouer le rôle `Acquéreur` dans l'association de type `Acquisition` au niveau instances.
- une association de type `type d'occurrence permis` reliant t_c et t_o permet à un topic de classe t_c de posséder une occurrence de type logique t_o ;

[1] Un système de pointeur extérieur au formalisme des TM permet cependant de créer des liens entre topics d'espaces différents. On peut par exemple associer un terme particulier du thesaurus à une ressource de l'espace d'annotation.

– une association de type `a pour type physique` reliant t_o et t_p spécifie que toute occurrence de type logique t_o a pour type physique t_p ce qui permet de contraindre la valeur de l'occurrence (donnée numérique, textuelle, format date, pointeur) et de l'interpréter correctement ;
– l'association de type `classe-sousclasse` définit un ordre partiel sur les topics et permet l'héritage des `types d'association permis` et des `types d'occurrence permis`.

2.3 Les services d'ITM

S'appuyant sur sa base de connaissances stockée dans un SGBD, ITM propose différents services d'acquisition et d'exploitation de ces connaissances.

L'acquisition des connaissances est réalisée par l'intermédiaire d'interfaces à base de formulaires "dynamiquement contrôlés" par l'ontologie correspondante au niveau modèle. Il existe par ailleurs un module d'annotation semi-automatique qui permet d'alimenter ITM avec la connaissance extraite de documents non structurés et semi-structurés sur la base de méthodes linguistiques [1]. Enfin un module d'importation de connaissances au format XTM (XML Topic Maps) permet d'intégrer des connaissances issues d'autres outils. En particulier, cela permet d'intégrer des ontologies de domaine construites à l'aide d'éditeurs spécialisés.

L'exploitation des connaissances peut se faire par navigation graphique dans le réseau Topic Maps. Une méthode de recherche "full-text" permet d'explorer les occurrences textuelles mais aussi les documents textes (externes à ITM) référencés par des topics. Une recherche multi-critères est également proposée. Elle permet, en spécifiant à l'aide de formulaires les caractéristiques de la structure recherchée, de retrouver :

– des topics particuliers en spécifiant un profil de topic : son type, les types d'occurrences possédées et les types d'association parmi lesquelles il joue un rôle ;
– des associations particulières (et les topics qui y participent) à partir de leur type, du type des rôles que l'on veut inclure et de profils pour les topics jouant ces rôles.

Ce mécanisme de recherche multi-critères permet donc de retrouver des sous-structures TM correspondant à un ensemble de topics reliés entre eux par une association.

Il est à noter que cet ensemble de services constitue une API qui peut être utilisée comme un service web.

2.4 Les besoins en raisonnement

Tout en étant un outil complexe de gestion des connaissances, ITM ne permet pas la mise en œuvre de raisonnements. On peut distinguer deux sortes de scénarios utiles à ITM nécessitant des capacités de raisonnement.

1. La maintenance de la base de connaissances : lorsqu'un ajout ou une modification de la base de connaissances est réalisé, on souhaite d'une part contrôler la cohérence des connaissances modifiées mais aussi propager les modifications effectuées à tous les niveaux inférieurs. Par exemple, l'ajout d'une occurrence obligatoire à un topic de l'ontologie de domaine doit entraîner l'ajout, la vérification ou la mise à jour de cette occurrence pour toutes les instances de ce topic dans la base d'annotations.
2. L'interrogation de la base de connaissances : on souhaite interroger ITM en utilisant un langage déclaratif de requêtage assez expressif pour pouvoir retrouver n'importe quelle sous-structure du réseau TM, tout en prenant en compte les connaissances ontologiques du niveau supérieur.

Par ailleurs, les besoins clients font apparaître l'intérêt de l'intégration d'un nouveau type de connaissances, les règles d'inférence, qui permettent de représenter des connaissances implicites. Citons comme usage simple des règles d'inférence la représentation de propriétés de relations, comme la symétrie ou la transitivité. Les règles pourront être prises en compte lors de l'ajout de connaissances dans la base pour mettre en œuvre un contrôle de cohérence sémantique ou lors de son interrogation.

Définir précisément les notions de cohérence d'une part, et de question/réponse d'autre part nécessite de doter les connaissances d'ITM d'une sémantique formelle. Dans ce but, nous avons fait le choix de définir une transformation des connaissances d'ITM dans un formalisme opérationnel de représentation de connaissances afin de bénéficier des résultats théoriques et logiciels existants.

3 la famille SG

Les graphes conceptuels, et en particulier la famille $\mathcal{SG}$ [3], nous semble être un "bon" formalisme de représentation de connaissances sur lequel bâtir les mécanismes de raisonnement envisagés. Différents arguments motivent ce choix :

– il existe une proximité évidente entre les topic maps et les graphes conceptuels simples ;
– les graphes conceptuels possèdent une sémantique formelle en logique des prédicats ;
– la famille $\mathcal{SG}$ fournit différents types de connaissances : les faits représentés par des graphes conceptuels simples, les règles d'inférence, ainsi que les contraintes (que nous n'utiliserons pas dans cet article). Les règles sont naturellement candidates à fournir les règles d'inférence dont nous avons besoin ;
– les raisonnements de la famille $\mathcal{SG}$ sont basés sur des opérations de graphe. Ils s'appuient directement sur les connaissances représentées, sans passer par un langage logique. Ceci fournit potentiellement des possibilités

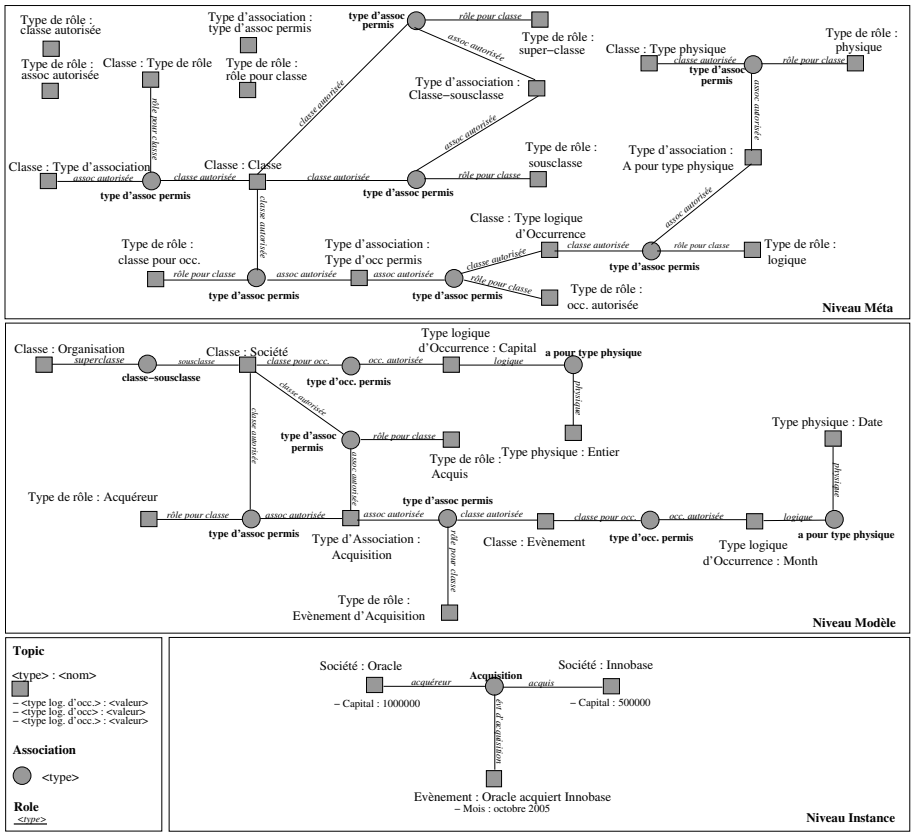

FIG. 3 – Une partie du modèle M_{ITM} avec son niveau méta, modèle et instances.

d'explication des raisonnements à l'utilisateur puisque les raisonnements peuvent être visualisés dans le formalisme qu'il connaît et directement sur les connaissances qu'il a définies. Précisons que les raisonnements sont adéquats et complets par rapport à la sémantique logique des connaissances représentées ;

– les algorithmes développés pour les graphes conceptuels, et en particulier pour la famille $\mathcal{SG}$, utilisent des techniques de combinatoire (notamment de théorie des graphes et de satisfaction de contraintes) qui les rendent particulièrement efficaces ;
– la famille $\mathcal{SG}$ est implémentée dans COGITANT, une API de développement d'applications à base de graphes conceptuels, utilisable librement [6].

Cette section présente de façon relativement informelle les graphes conceptuels de base. Pour plus de précision, voir par exemple [4] pour une définition du modèle de base ou [3] pour une présentation de toute la famille $\mathcal{SG}$.

3.1 Les graphes simples

Un *graphe conceptuel simple (SG)* est un graphe biparti étiqueté : l'une des classes de sommets, dite de sommets *concepts*, représente des entités, et l'autre, dite de sommets *relations*, représente des relations entre ces entités ou des propriétés de ces entités. Le graphe G_1 de la Figure 4 peut être vu comme représentant la connaissance suivante : "un grand groupe de l'agroalimentaire acquiert la société $S1$; $S1$ qui contrôle la société $S2$, elle aussi spécialisée dans l'agroalimentaire".

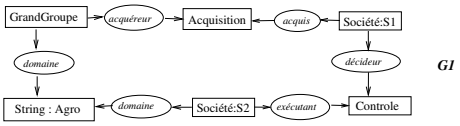

FIG. 4 – Un SG : les relations étant toutes binaires dans ces exemples, la numérotation des arêtes a été remplacée par des flèches. Le marqueur générique n'est pas représenté.

Les étiquettes des sommets sont prises dans un vocabulaire appelé **support** qui peut être plus ou moins riche. Nous considérerons ici un support comme une structure $S = (T_C,\ T_R,\ I,\ \sigma)$. T_C est un ensemble de types de concepts et T_R est un ensemble de relations d'arité (nombre d'arguments) quelconque. T_C et T_R sont partiellement ordonnés, l'ordre partiel traduisant une relation de spécialisation ($t' \leq t$ s'interprète par "t' est une spécialisation de t"). I est un ensemble de marqueurs dits individuels. σ associe à toute relation une signature qui définit le type maximal de chacun de ses arguments. Le support peut être vu comme

une ontologie rudimentaire et les SG encodent des connaissances assertionnelles (que nous appellerons des *faits*) : ils assertent l'existence d'entités et de relations entre ces entités.

Un sommet concept d'un SG est étiqueté par un couple $t : m$ où t est un type de concept et m est un marqueur. Si le sommet représente une entité non identifiée, ce marqueur est le marqueur générique $*$ (le sommet est dit *générique*), sinon le marqueur est un élément de I (le sommet est dit *individuel*). Dans le graphe G_1 de la figure 4 par exemple, le sommet `[Société :S1]` désigne "la" société S1, tandis que le sommet `[GrandGroupe :*]` désigne "un" grand groupe. Un sommet relation est étiqueté par une relation r et, si n est l'arité de r, n arêtes sont incidentes à ce sommet ; ces arêtes sont totalement ordonnées. De façon classique, dans les dessins, les sommets concepts sont représentés par des rectangles et les sommets relations par des ovales, et l'ordre sur les arêtes incidentes à un sommet relation n-aire par une numérotation de ces arêtes de 1 à n. On note $G = (C_G, R_G, E_G, l_G)$ un SG où C_G est l'ensemble des sommets concepts, R_G l'ensemble des sommets relations, E_G l'ensemble des arêtes et l_G la fonction d'étiquetage des sommets et des arêtes.

3.2 Sémantique et raisonnement

La notion fondamentale pour comparer des SG est une application d'un SG dans un autre appelée *projection* (un homomorphisme de graphes en termes de théorie des graphes). Intuitivement, l'existence d'une projection de G dans H montre que la connaissance représentée dans G est contenue dans H. Une projection π de G dans H est plus précisément une application de C_G dans C_H et de R_G dans R_H qui conserve les arêtes (si on a une arête entre r et c étiquetée i dans G alors on a une arête entre $\pi(r)$ et $\pi(c)$ étiquetée i dans H) et peut "spécialiser" les étiquettes des sommets.

La figure 5 montre un exemple de projection d'un graphe Q sur un graphe G_2 (sachant que `GrandGroupe < Société`). Les sommets grisés donnent l'image de la projection (sur cet exemple la projection est injective).

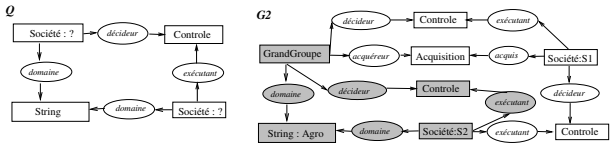

FIG. 5 – Image de la projection de Q dans G_2

Ce formalisme de base est muni d'une sémantique en logique du premier ordre par le biais d'une application notée Φ [11]. Les types de concepts sont traduits par des prédicats unaires, les relations par des prédicats de même arité, et les marqueurs individuels par des constantes. Au support S on associe un ensemble de formules $\Phi(S)$ traduisant les relations de spécialisation sur les types de concepts et les relations : si t et t' sont des types de concept, avec $t' < t$, on a la formule $\forall x t'(x) \rightarrow t(x)$; similairement si r et r' sont des relations n-aires, avec $r' < r$, on a la formule $\forall x_1 \ ... \ x_n r'(x_1 \ ... \ x_n) \rightarrow r(x_1 \ ... \ x_n)$. Un SG (soit G) asserte l'existence d'entités et de relations entre ces entités. Il se traduit naturellement par une formule $\Phi(G)$ positive et conjonctive, fermée existentiellement, où chaque sommet concept est traduit soit en la constante associée à son marqueur individuel, soit en une nouvelle variable s'il est générique. La formule associée au graphe G_1 de la figure 4 est : $\Phi(G_1) = \exists x \exists y \exists z GrandGroupe(x) \wedge Acquisition(y) \wedge Controle(z) \wedge Societe(S1) \wedge Societe(S2) \wedge String(Agro) \wedge acquereur(x, y) \wedge acquis(S1, y) \wedge domaine(x, Agro) \wedge domaine(S2, Agro) \wedge decideur(S1, z) \wedge executant(S2, z)$.

Le résultat fondamental d'adéquation et de complétude de la projection établit l'équivalence entre l'existence d'une projection et la déduction logique sur les formules associées aux SG ([11] pour l'adéquation et [4] pour la complétude). Précisons que pour obtenir la complétude, il est nécessaire que H soit sous une forme dite normale : tout marqueur individuel apparait au plus une fois dans H (autrement dit, il n'existe pas deux sommets qui représentent la même entité identifiée). Cette restriction peut-être levée en considérant une variante de la projection [5].

Considérons une base de connaissances composée d'un support et d'une base F de faits. Cette base peut-être interrogée par le mécanisme de projection. Sous sa forme la plus simple, une requête est elle-même un SG (Figure 5 : vu comme une requête, le SG Q demande à trouver le motif suivant "deux sociétés d'un même domaine tels que l'une contrôle l'autre". La base répond à la requête s'il existe une projection de Q dans G_2, ce qui se traduit par le fait que Q se déduit de la base de connaissances. Toute projection de Q dans G_2 peut-être vue comme une réponse à la requête. Une requête peut aussi comporter des sommets (concepts génériques) distingués. En ce cas seuls ces sommets sont considérés pour construire une réponse à la requête (Figure 5 : si les 2 sommets [Société] sont distingués, la requête demande les *couples de Sociétés* d'un même domaine tels que l'une contrôle l'autre. Classiquement ces sommets sont marqués par le signe ' ?'. Une requête de cette forme est alors équivalente à une requête conjonctive de base de données.

3.3 Les règles de SG

A ce formalisme "noyau" la famille $\mathcal{SG}$ ajoute deux types de connaissance : les *règles* et les *contraintes* (ce dernier type de connaissances n'étant pas abordé dans cet article).

Une **règle** d'inférence exprime une connaissance de la forme "si hypothèse alors conclusion", où hypothèse et conclusion sont deux SG. La figure 6 présente deux exemples de règles. Les pointillés lient certains sommets de l'hypothèse et de la conclusion ; ces sommets sont ap-

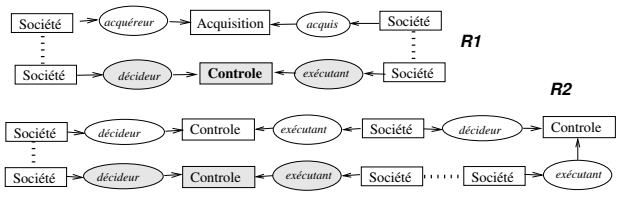

FIG. 6 – Règles

pelés sommets *frontières* ; les sommets de la conclusion autres que les sommets frontières apparaissent en grisé. Les règles $R1$ et $R2$ expriment des propriétés du type de concept "contrôle" : l'acquisition d'une société entraine le contrôle de cette société ($R1$) et le contrôle s'exerce de facon "transitive" ($R2$).

Une règle R s'applique à un SG F s'il existe une projection de l'hypothèse de R dans F. L'application de R à F selon une telle projection π consiste à "attacher" à F la conclusion de R, en fusionnant chaque sommet frontière de la conclusion avec l'image par π du sommet frontière lui correspondant dans l'hypothèse. Figure 4 : en appliquant $R1$ au SG G_1 de la figure 4 (avec `GrandGroupe` $<$ `Société`) puis $R2$ sur le graphe obtenu, on obtient le SG G_2 de la figure 5.

Notons que ce mécanisme est complètement basé sur des opérations de graphe. Etant donné un fait F et un ensemble de règles $\mathcal{R}$, on dit qu'un SG F' se dérive de F par $\mathcal{R}$ s'il existe une séquence d'applications de règles de $\mathcal{R}$ menant de F à F'.

La formule associée à une règle R est de la forme $\Phi(R) = \forall x_1 \ldots x_p \ ((hyp) \rightarrow \exists y_1 \ldots y_q \ (conc))$, où $x_1 \ldots x_p$ et $y_1 \ldots y_q$ sont les variables respectivement associées aux sommets de l'hypothèse et aux sommets de la conclusion, hyp et $conc$ sont les conjonctions d'atomes respectivement associés à l'hypothèse et à la conclusion.

Une base de connaissances K est maintenant composée du support, d'un ensemble de règles R et d'un ensemble de faits F. Le mécanisme d'interrogation doit donc prendre en compte la connaissance implicite encodée dans les règles. La base de connaissances répond à une requête Q s'il existe un SG F' dérivé de F (par les règles de R) tel que Q se projette dans F'. Considérons par exemple les figures 4, 5 et 6 : la base formée du SG G_1 et des règles $R1$ et $R2$ répond à la requête Q ; en effet Q se projette dans G_2 dérivé de G_1 par les règles. On dispose de mécanismes de chaînage avant et de chaînage arrière adéquats et complets par rapport à la déduction logique [10].

Précisons que le problème d'interrogation n'est plus décidable (autrement dit il n'existe pas d'algorithme qui résolve ce problème en un temps fini quelles que soient les données) si l'on ne restreint pas la forme des règles. On peut notamment s'assurer que les règles utilisées permettent de "saturer" la base de faits (c'est-à-dire d'appliquer les règles de toutes les façons possibles en un temps fini) ; répondre à une requête consiste alors à rechercher les projections de la requête dans la base saturée. Une alternative à la saturation de la base consiste à utiliser le chaînage arrière.

4 Un moteur graphes conceptuels pour ITM

Cette section détaille la transformation des TM d'ITM vers les SG et montre comment elle permet d'enrichir ITM. Comme toute la connaissance de ITM est représentée en TM, plusieurs choix de transformation sont possibles :

1. Traduire le méta-modèle ITM (en fait le formalisme TM) dans un support SG et traduire les niveaux inférieurs comme une base de faits SG. Cela permet de traiter les connaissances d'ITM en toute généralité mais cela exploite mal les spécificités de représentation du formalisme SG.
2. Plaquer le méta-modèle ITM sur le formalisme SG, traduire le niveau modèle dans le support et traduire le niveau instances comme une base de faits SG.
3. N'appliquer le choix 2 qu'à l'espace de travail dédié aux annotations en considérant que seul cet espace concerne la représentation de connaissances, le thésaurus contenant des connaissances linguistiques et les autres espaces étant dédiés à l'organisation des documents.

C'est ce dernier choix que nous présentons dans cette section.

4.1 Transformation d'une base ITM vers une base SG

Nous notons f la transformation 3 d'une base ITM vers les SG, qui code l'ontologie de domaine en un support et la base d'annotations en un SG (ou ensemble de SG). La figure 7 présente le résultat de la transformation f sur les TM de la figure 3.

Transformation de l'ontologie vers le support

1. Trois types de concept C, TA et TPO sont créés comme sous-type d'un type universel $\top$. Ces sous-types représentant respectivement le supertype des classes de topic, types d'association et types physiques d'occurrence. Tous les topics t dont le $type$ est `classe`, `type d'association` ou `type physique` deviennent respectivement par f des types de concepts sous-types de C, TA et TPO et sont identifiés par $nom(t)$. L'ordre partiel sur T_C est induit par celui de l'association `classe-sousclasse`. Par exemple, les topics `Société` et `Organisation` (du niveau modèle) deviennent par f des éléments de T_C ordonnés de la manière suivante *Société* $\leq$ *Organisation* $\leq C$.

2. Deux relations TR, TLO, ayant respectivement pour signature $\sigma(TR) = (C, TA)$ et $\sigma(TLO) = (C, TPO)$, sont créées pour représenter respectivement les supertypes des rôles et des types logiques d'occurrence. Tous les topics t dont le $type$ est `type de rôle` qui participent à une association ternaire de $type$ `type d'association permis` avec des topics c de $type$ `classe` et ta de $type$ `association` deviennent par f des relations identifiées par $nom(t)$ sous-relations de TR et ont pour signature $\sigma(nom(t)) = (f(c), f(ta))$. Tous les topics t dont le $type$ est `type logique d'occurrence` sont reliés par une association de $type$ `a pour type physique` à un topic de tpo et par une association de $type$ `type d'occurrence permis` à un topic c; ils deviennent par f des relations identifiées par $nom(t)$ sous-relations de TLO et ont pour signature $\sigma(nom(t)) = (f(c), f(tpo))$. Par exemple, le topic (du niveau modèle) `acquis` relié par `type d'association permis` à `Société` et `Acquisition` devient par f une relation *acquis* mise sous TR et ayant pour signature *σ(acquis)=(Société, Acquisition)*.

Transformation de la base d'annotations en SG

1. Tous les topics t dont le nom est spécifié sont transformés par f en un sommet concept individuel étiqueté $f(type(t)) : nom(t)$; de plus $nom(t)$ est inséré dans l'ensemble des marqueurs individuels I. E.g. le topic `Oracle` de $type$ `Société` (du niveau instances) est transformé par f en un sommet concept individuel d'étiquette *Société :Oracle*.
2. Tous les topics t n'ayant pas de nom spécifié (resp. les associations a) sont transformés par f en un sommet concept générique étiqueté $f(type(t)) : *$ (resp. $f(type(a)) : *$). E.g. l'association de $type$ `acquisition` (du niveau instances) est transformée par f en un sommet concept générique d'étiquette *Acquisition :**.
3. Chaque arête e reliant une association a à un topic t est transformée par f en un sommet relation d'étiquette $f(type(e))$ dont les deux arêtes incidentes étiquetées resp. 1 et 2 le relient resp. à $f(t)$ et $f(a)$. E.g. l'arête portant le rôle `acquéreur` est transformée par f en un sommet relation d'étiquette *acquéreur* le reliant aux deux sommets concepts précédents.
4. Chaque occurrence $o = (v, tpo, tlo)$ d'un topic t engendre par f un sommet relation d'étiquette $f(tlo)$ dont les deux arêtes incidentes étiquetées resp. 1 et 2 le relient resp. au sommet concept $f(t)$ et à un nouveau sommet concept individuel étiqueté $f(tpo) : v$. E.g. l'occurrence `1000000` du topic `Oracle` de type logique `capital` et de type physique `Entier` devient par f un sommet concept d'étiquette *Entier :1000000* relié au sommet concept *Société :Oracle* par un sommet relation *capital*.

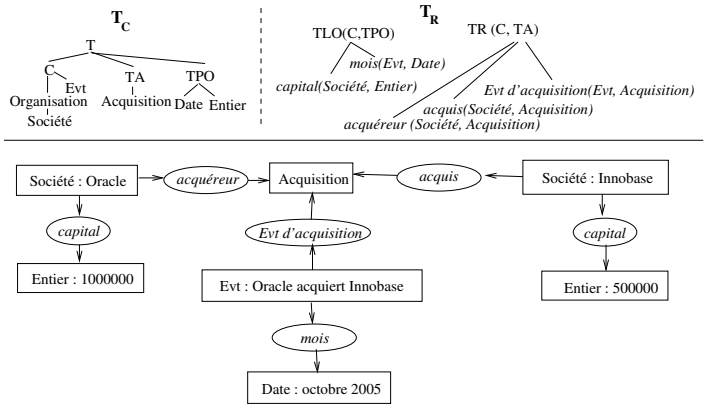

FIG. 7 – Résultat de la transformation de la TM en figure 3

4.2 Propriétés de cette transformation

La fonction f est un codage, au sens où elle est injective, ce qui la rend *réversible*. Ceci est primordial pour que les connaissances inférées puissent être restituées à ITM. De plus, elle respecte la sémantique opérationnelle de ITM, puisque l'association *superclasse-sousclasse* traduite comme ordre dans le support et les associations `type d'association permis`, `type d'occurrence permis` et `a pour type physique` sont reprises comme signatures de relations dans le support.

Quels sont les apports de la transformation f à ITM ? D'une part elle dote indirectement ITM d'une sémantique formelle, celle des graphes conceptuels. D'autre part, elle permet d'enrichir immédiatement ITM sur deux points : la définition d'un langage déclaratif de requêtes et l'intégration à ce mécanisme de règles d'inférence. Du fait de la propriété de réversibilité de f, ces enrichissements sont immédiats.

ITM possède des fonctionnalités de requêtage qui permettent de retrouver des sous-structures du réseau limitées à un topic ou à une association et ses voisins. Les SG possèdent un mécanisme naturel d'interrogation basé sur la projection, qui permet de rechercher une sous-structure *quelconque* du réseau. En étendant la syntaxe des TM pour intégrer le signe ' ?', nous obtenons un langage de requêtes pour ITM. Il est important de noter que, du fait de la réversibilité de f, toute réponse à la requête (que l'on considère la projection elle-même ou le sous-graphe image de la requête par la projection) peut être traduite en une réponse sur les TM.

Précisons toutefois que les prédicats associés aux types physiques des occurrences (opérateurs de comparaison no-

tamment) ne sont pas encore pris en compte dans ce langage de requêtes.

Les règles quant à elles permettent de prendre en compte des axiomes ontologiques plus riches que l'association *superclasse-sousclasse* telles que la transitivité de certains types d'association (cf. [8]) et peuvent permettre de compléter a posteriori le réseau d'instances d'une base ITM dans un objectif d'enrichissement ou de détections d'incohérences.

5 Implémentation et perspectives

Nous avons décrit la transformation permettant de doter le système de gestion de connaissances ITM de raisonnements basés sur des opérations de graphes. Un premier prototype montrant la faisabilité de la démarche a été programmé. Il permet de transformer (par f) et charger dans CoGITaNT la connaissance d'ITM (ontologie de domaine et base d'annotations), puis de l'interroger. Cette section présente l'architecture globale (il s'agit de l'architecture à terme, non encore complètement implémentée par le prototype), puis les premières expérimentations effectuées.

5.1 Architecture

CoGITaNT est une API fournissant des composants logiciels pour développer en C++ des moteurs d'inférence compatibles avec la famille $\mathcal{SG}$ [6]. Elle fournit aussi un moteur d'inférence prêt à l'emploi sous forme d'un serveur TCP/IP auquel on adresse des requêtes dans un format spécifique XML, le CoGXML. Ainsi un client peut être programmé en un tout autre langage et bénéficier des services inférentiels du serveur CoGITaNT. A terme, l'architecture globale intégrera le serveur de CoGITaNT et ITM en tant que client de ce dernier de telle sorte qu'à partir d'ITM on puisse envoyer :

- des directives de chargement dans CoGITaNT de la base de connaissances exportée en CoGXML ;
- des directives d'ajout et de suppression de connaissances synchronisées avec les modifications faites dans ITM ;
- des requêtes sur le réseau TM dont la structure est trop complexe pour être directement réalisée dans ITM, ou nécessitant la prise en compte de règles ;
- des directives de vérification de contraintes.

La figure 8 présente la base de connaissances de ITM entourée d'une API qui permet son exploitation et la communication avec CoGITaNT. L'essentiel des raisonnements et requêtes qui seront gérés par CoGITaNT portent sur la base d'annotations. Une API d'export fournit sous forme de fichier CoGXML le graphe et le support correspondants à la base d'annotations et la portion d'ontologie concernée. Au démarrage, CoGITaNT charge le support et le graphe en mémoire. Si des modifications ultérieures sont effectuées

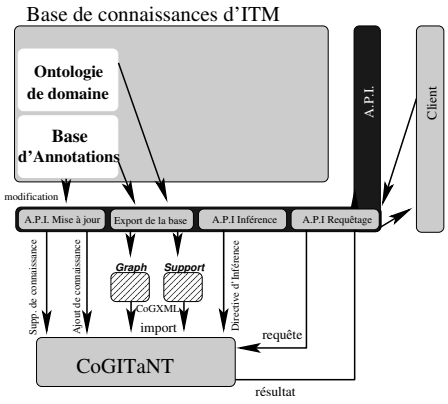

FIG. 8 – L'architecture entre ITM et CoGITaNT

dans ITM, l'API les détecte et met à jour en temps réel la base de connaissances de CoGITaNT. Lors de mises à jour, ITM peut lancer dans CoGITaNT des applications de règles jusqu'à saturation. Lorsqu'un client formule une requête, ITM la traduit en SG et la fournit à CoGITaNT. Ce dernier la projette sur sa base saturée de connaissances inférées et retourne le résultat à ITM.

5.2 Expérimentations

Un premier prototype nous a permis d'effectuer quelques expérimentations dont nous présentons ici les résultats. La base de connaissances ITM utilisée relève du domaine de l'intelligence économique. La transformation de cette base dans le formalisme SG a été réalisée en pré-traitement "off-line" du service. Cette transformation a fourni un SG de 6500 sommets concepts et 5500 relations. La machine utilisée est un AMD Athlon 1800+ 1.53 Ghz 768 Mo RAM.

Le temps de chargement de la transformation de la base stockée en CoGXML dans le moteur d'inférence est de 4 min 32 secondes. Nous avons tout d'abord comparé les temps de réponse des services actuels d'ITM avec le nouveau service sur des requêtes simples contenant une seule association (les seules réalisables directement par ITM). Nous avons ensuite testé des requêtes plus complexes avec ou sans cycle. Les tableaux 9 pour CoGITaNT et 10 pour ITM présentent ces résultats. Seules les deux premières lignes du tableau 9 sont comparables au tableau 10 ; elles correspondent à des requêtes simples avec les nom des topics recherchés ou sans les nom des topics recherchés (donc correspondant à des concepts génériques). Précisons que les requêtes fournies au moteur d'inférence ne portaient que sur la partie symbolique du réseau de connaissances et non sur les valeurs des occurrences.

Ces premiers résultats encourageants montrent que notre approche est réaliste puisque les temps de réponse sont identiques à ceux d'ITM, ou meilleurs. Des tests plus poussés nous permettront d'étudier l'influence du nombre de ré-

Nb requêtes	Nature graphique	Nb génériques	Nb indiv	Nb relations	Nb réponses	Tps de réponses
5	chaine	2	1	2	4,5,0,2,1	0.3 sec
2	chaine	3	0	2	686,74	2 sec, 0.3 sec
4	arbre	5	0	4	202,4,11,33	2.2 s, 0.3 s, 0.3 s et 0.4 s
1	arbre	7	0	6	1	0.3 s
1	cycle	4	0	4	3	2 s

FIG. 9 – Evaluation du requêtage à l'aide de CoGITaNT.

Nb requêtes	Nature graphique	Nb topics + assocs	Nb relations	Nb réponses	Tps de réponses
5	chaine	3	2	4,5,0,2,1	0.3 sec
2	chaine	3	2	686,74	18 sec,2 sec

FIG. 10 – Evaluation du requêtage dans ITM.

ponses, de la structure de la requête et de la base, et de la structure de l'ontologie sur les performances du moteur.

Une expérimentation d'utilisation des règles a également été menée. Les règles R_1 et R_2 ont été apliquées successivement jusqu'à saturation. La règle $R1$ de la figure 6 a été appliquée en une seule passe et en un délai de 0.1 secondes pour un nombre de 77 projections, tandis que pour saturer la base à partir de la règle $R2$ il a fallu deux passes chacune d'un délai de 0.2 et 0.3 secondes pour un nombre respectif de 17 et 4 projections. Après ces applications, la projection d'une requête de détection de symétrie pour l'association `contrôle` a permis de découvrir en 1.9 secondes 3 sous-graphes de la base où l'association `contrôle` formait un cycle, et de corriger ces incohérences. Cela montre les possibilités d'utilisation du moteur d'inférence dans un scénario de validation de la base. Il faut par ailleurs noter qu'un mécanisme plus complexe de contraintes existe dans la famille $\mathcal{SG}$ et pourra être exploité.

5.3 Traitement des attributs

Les occurrences de ITM sont représentées par des attributs numériques ou textuels dans un SGBD qui fournit des opérateurs de comparaison adaptés pour les interroger : requêtes du type "retourne tous les topics dont la valeur d'une occurence de type entier est comprise entre deux nombres" ou "retourne tous les topics dont le nom contient telle sous-chaine". Comme illustré par la transformation f, la représentation de ces attributs dans un SG est possible. Mais elle ne semble pas pertinente car leur nature diffère de celle des sommets concepts et relations. En effet, les attributs n'ont pas la même sémantique formelle, leur taille en mémoire est variable et ils utilisent très peu les avantages d'une représentation réseau puisque leurs valeurs, souvent différentes, sont représentées par un ajout de sommets pendants. Une représentation hybride SGBD/GC semble être un bon compromis pour obtenir un formalisme de représentation des connaissances qui conjugue les avantages d'une modélisation par attribut-valeur à ceux d'une représentation par graphes conceptuels. La prochaine étape de l'intégration dans ITM du moteur d'inférence à base de graphes conceptuels devrait aboutir à l'élaboration d'un tel système.

Références

[1] F. Amardeilh, P. Laublet et J.L. Minel. Annotation documentaire et peuplement d'ontologie à partir d'extractions linguistiques. In *Acte IC*, pages xx–yy, 2005.

[2] P. Auillans, P. Ossana de Mendez, P. Rosenstiehl et B. Vatant. A formal model for topic maps. In I. Horrocks et J. Hendler, éditeurs, *Proc. of First International Semantic Web Conference (ISWC'02)*, volume 2342, pages 69–83, 2002.

[3] J.-F. Baget et M.-L. Mugnier. Extensions of Simple Conceptual Graphs : The Complexity of Rules and Constraints. *JAIR*, 16 :425–465, 2002.

[4] M. Chein et M.-L. Mugnier. Conceptual Graphs : Fundamental Notions. *Revue d'Intelligence Artificielle*, 6(4) :365–406, 1992.

[5] M. Chein et M.-L. Mugnier. Types and Coreference in Simple Conceptual Graphs. In *Proc. ICCS'04*, LNAI. Springer, 2004.

[6] CoGITaNT. Conceptual graphs integrated tools allowing nested typed graphs. http ://cogitant.sourceforge.net, 1997.

[7] O. Corby, R. Dieng-Kuntz et C. Faron-Zucker. Querying the Semantic Web with Corese Search Engine. In *Proceedings of 3^{rd} Prestigious Applications Intelligent Systems Conference (PAIS) Valence*, pages 705–709, 2004.

[8] F. Furst, M. Leclère et F. Trichet. Operationalizing domain ontologies : a method and a tool. In *Proceedings of the 16th ECAI*, pages 318–324. IOS Press, 2004.

[9] ISO/IEC :13250. Topic maps (2000). http ://www.y12.doe.gov/sgml/sc34/document/0129.pdf.

[10] E. Salvat et M.-L. Mugnier. Sound and Complete Forward and Backward Chainings of Graph Rules. In *Proc. of ICCS'96*, volume 1115 of *LNAI*, pages 248–262. Springer, 1996.

[11] J. F. Sowa. *Conceptual Structures : Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[12] TopicMaps.Org. Xml topic maps (xtm) 1.0. http ://www.topicmaps.org/xtm/1.0/, 2001.