République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abou Backr Belkaid - Tlemcen
Faculté des sciences

Mémoire

Présenté en vue de l'obtention du diplôme de

Magister en informatique

Option : Intelligence Artificielle et Aide a la Décision

ETUDE ET RÉALISATION D'UN RAISONNEUR DANS LES LOGIQUES DE DESCRIPTIONS

Présenté par Mr: **BOURZIG Sofiane** Dirigé par : **Dr. LEHIRECHE Ahmed**

Soutenue-le : … /07 /2011 devant le jury composé de:

Président :
Dr. CHIKH Mohammed Amine **Maitre de conférences classe A**
Université Abou Backr Belkaid - Tlemcen

Examinatrice :
Dr. DIDI Fedoua **Maitre de conférences classe A**
Université Abou Backr Belkaid - Tlemcen

Invité :
Dr. BENAMMAR Abdelkarim **Maitre de conférences classe B**
Université Abou Backr Belkaid - Tlemcen

Dédicaces

Je dédie humblement ce mémoire…

Pour ton amour, ton affection et ton soutien. Pour ton courage et ton sacrifice. Je te dédie chère et tendre mère, un résultat modeste de ta bienveillance et tes longues années de patience.

A la mémoire de mon père.

Remerciement

Je remercie Allah.

Je ne remercierai jamais assez ma chère mère, sans lui, rien n'aurait été possible. Leur amour et leur dévotion, a fait de moi ce que je suis. Elle m'a vus naître puis grandir, et n'a jamais rechigné à apporter tout ce qu'elle peut, de me supporter et de me réconforter. Merci ma mère.

Je remercie aussi mon encadreur Mr. LEHIRECHE Ahmed Maitre de conférence classe A à l'UDL de sidi belabbes qui m'a guidé avec patience et gentillesse et m'a fait profiter de sa grande expérience ainsi que de ses précieuses remarques qui ont grandement contribué à améliorer la qualité de ce mémoire. Qu'elle soit ici assurée de mon très grand respect.

Je tiens à remercier très sincèrement l'ensemble des membres du jury qui me font le grand honneur d'avoir accepté de juger mon travail.

Enfin, je remercie tous mes amis et mes collègues, sans exception et sans distinction, pour leur présence et leur soutien. Merci à tous.

Résume

Un système à base de connaissances est un programme capable de raisonner sur un domaine d'application pour résoudre un certain problème, en s'aidant de connaissances relatives au domaine étudié. Les connaissances du domaine sont représentées par des entités qui ont une description syntaxique à laquelle est associée une sémantique. Les logiques de descriptions est le nom le plus récent d'une famille d'un formalisme de représentation de connaissance d'un domaine d'application. Dans ce formalisme, un concept permet de représenter un ensemble d'individus, tandis qu'un rôle représente une relation binaire entre individus. Un concept correspond à une entité cognitive générique d'un domaine d'application. Un individu correspond à une entité particulière dite instance d'un concept. Une description est une spécification d'un ensemble de termes ou chaque terme représente un concept du domaine. Notre mémoire introduit, comme études théoriques, les fondements des logiques de descriptions : la syntaxe et la sémantique d'un concept et de relation conceptuelle, la relation de subsomption et le raisonnement terminologiques. Il existe deux approches principales pour appréhender la détection des relations de subsomption et de raisonnement terminologique. La première approche s'appuie sur les algorithmes de normalisation comparaison : NC ; La seconde approche sur la méthode des tableaux sémantiques appliquée aux logiques de descriptions. Pour la mise en œuvre de notre raisonneur DL, nous avons implanté la méthode NC en tant qu'expérimentation. Nous avons, finalement, opté pour la méthode des tableaux sémantiques.

Mots-clés : représentation de connaissances, logiques de descriptions, TBox, ABox, raisonnement, subsomption, algorithmes de tableaux.

Abstract

A knowledge-based system is a program that can reason about an application domain to solve a certain problem, with the help of knowledge of the area studied. The domain knowledge is represented by entities that have a syntactic description that is associated semantics. Descriptions logics is the latest name of a family of knowledge representation formalism that represents the knowledge of an application domain. In this formalism, a concept used to represent a set of individuals, while a role represents a binary relation between individuals. A concept is a cognitive and generic entity of a domain. An individual is a particular entity said instance of concept. A description is a specification of a set of terms where each term represents a concept of domain. Our Memory introduces, as theoretical studies, the foundations of description logics: syntax and semantics of a concept, and of conceptuelle relation, the relation of subsumption and reasoning terminology. There are two main approaches to understand the detection of relationships of subsumption and terminological reasoning. The first approach is based on algorithms for Standardization comparison : NC ; The second approach is based on the method of tableau algorithms applied to description logics. For the implementation of our DL reasoner, we implemented the NC method as experiment. We finally opted for the method of semantic tableaux.

Keywords: knowledge representation, descriptions logics, TBox, Abox, reasoning, subsumption, tableau algorithms

ملخص:

النظام القائم على قاعدة معلومات هو برنامج قادر على إيجاد الحلول لمشاكل مجال التطبيق, وذلك بمساعدة المعلومات المتعلقة بالمجال المدروس.معلومات المجال تمثل بواسطة دلالات لها وصف نحوي كل منها تملك معنى معين. منطق الوصف هو الاسم الأحدث لعائلة تمثيل المعلومات.المفهوم يسمح بتمثيل مجموعة أفراد, بينما العلاقة تمثل الرابط بين الأفراد.المفهوم عبارة عن دلالة عامة و إدراكية لمجال التطبيق. الفرد عبارة عن عينة خاصة للمفهوم. الوصف هو تحديد لمجموعة من المصطلحات حيث يمثل كل مصطلح مفهوم المجال. مذكرتنا تعرض كدراسة نظرية: أسس منطق الوصف, كيفية بناء المفاهيم و العلاقات و معانيهما, علاقة الاحتواء بين المفاهيم (subsomption) و التفكير بالمصطلحات. هناك نوعان من الأساليب الرئيسية لفهم و إيجاد علاقة subsomption والتفكير بالمصطلحات: النهج الأول يعتمد على خوارزميات المقارنة بالتوحيد, النهج الثاني على أسلوب اللوحات الدلالية التي تنطبق على منطق الوصف. لتنفيذ المفكر الخاص بنا، طبقنا الأسلوب الأول كتجربة. اخترنا أخيرا أسلوب اللوحات الدلالية

Table des matières

Table des figures

Introduction générale :

Depuis ces tout débuts l'Intelligence Artificielle essaie de concevoir des systèmes qui ont le pouvoir de simuler la connaissance à travers l'utilisation de mécanismes de raisonnement automatique. Ces mécanismes font généralement appel à des bases de connaissances (BC), qui sont un ensemble de termes et d'axiomes permettant de contraindre l'interprétation et d'inférer de nouvelles informations. L'étude du développement et des propriétés de ces BC a donné naissance à tout un domaine de recherche, nommé l'Ingénierie de connaissances (IC).

Dans notre thèse nous allons aborder les bases de connaissances basées sur les logiques de descriptions qui permettent de représenter les connaissances relatives à un domaine de référence à l'aide de « description » qui peuvent être des concepts, des rôles et des individus. Les concepts modélisent des classes d'individus et les rôles des relations entre classes. Une sémantique est associée aux descriptions par l'intermédiaire d'une fonction d'interprétation.

Le choix de logiques de descriptions, se justifie par les points suivants:

- La plupart des logiques de descriptions sont décidables.
- Ils s'agissent d'un langage utilisé pour structurer des informations en utilisant la notion de concepts et de rôles.
- Actuellement, on dispose d'un nombre considérable de logiques de description qui varient en terme d'expressivité en utilisant certains constructeurs.

  De plus, des travaux ont consisté a déterminer la complexité correspondante et par conséquent, l'utilisateur est capable de choisir la logique qui répond a ses besoins en termes d'expressivité avec un moindre cout en termes de complexité.
- De nombreux raisonneurs DL ont été développés a l'instar de Fact ++ **[HOR 98].** La plupart de ces derniers utilisent des techniques d'optimisation sophistiquées. Par conséquent, ces raisonneurs sont général efficaces en pratique notamment sur des problèmes réels.
- Enfin, de nombreux langages de description d'ontologies sont basés sur les DLs étant données les caractéristiques précédentes. A titre d'exemple, on peut citer DAML+OIL **[HOR 02]** et le fameux langage OWL **[BER 08].**

L'accent au sein des LD est mis sur les services de raisonnement et plus particulièrement sur ceux pour la prise de décision : l'objectif principal des DL consiste à pouvoir raisonner efficacement pour minimiser les temps de réponse. Par conséquent, la communauté scientifique a publié de nombreuses recherches qui portent sur l'étude du rapport expressivité/performance des différentes DL **[FRA 03].** La principale qualité des DL réside dans leurs algorithmes d'inférence dont la complexité est souvent inférieure aux complexités des démonstrateurs de preuves de la logique de premier ordre **[TSA 03].**

Le raisonnement permet d'inférer des connaissances représentées implicitement à partir des autres contenues explicites dans les bases de connaissances.

Les éléments principaux qui caractérisent les logiques de descriptions sont le langage de description des concepts et des rôles, l'interprétation associée aux expressions du langage et la relation de subsomption.

La subsomption est la relation fondamentale existant entre les descriptions.

Pour tester les relations de subsomption entre les descriptions, deux voies ont été principalement explorées jusqu'à présent : les algorithmes de type normalisation-comparaison, abrégés en algorithmes NC, et une méthode dérivée de la méthode des tableaux sémantiques en logique classique **[GOC 91],** que nous appelons plus simplement dans la suite méthode des tableaux sémantiques.

Notre travail consiste à étudier le raisonnement au sein des logiques de description, en explorant les différents problèmes d'inférences, les taches de raisonnement ainsi que les différents algorithmes adoptés pour raisonner sur les DLs.

La suite de cette thèse est structurée de la façon suivante. Le premier chapitre et puisque Les DL utilisent une approche ontologique présente tout d'abord la notion d'ontologie, il met l'ontologie dans leur place au sein du processus de représentation des connaissances, en suite il présente leurs différents constituants, enfin illustre l'objectif d'intégration des ontologies dans les SBC.

Le deuxième chapitre de cette thèse est une introduction aux logiques de descriptions. En premier lieu, il présente le formalise de base des DL, en suite la modélisation des connaissances d'un domaine avec les DL qui se réalise en deux niveaux. Le premier, le niveau terminologique ou TBox, décrit les connaissances générales d'un domaine alors que le second, le niveau factuel ou ABox, représente une configuration précise.

Une TBox comprend la définition des concepts et des rôles, alors qu'une ABox décrit les individus en les nommant et en spécifiant en termes de concepts et de rôles, des assertions qui portent sur ces individus nommés. Plusieurs ABox peuvent être associés à une même TBox ; chacune représente une configuration constituée d'individus, et utilise les concepts et rôles de la TBox pour l'exprimer.

Le troisième chapitre aborde les services d'inférence usuels pour les LD, qui s'effectuent au niveau terminologique ou factuel en présentant les différentes taches de raisonnement pour la TBox ainsi que pour la Abox, Ensuite nous allons présenter deux types des algorithmes de raisonnement dans les DL à savoir les algorithmes structurels et les algorithmes logiques.

Le chapitre contribution consiste à exposer notre application, qui est développée avec le langage java, pour cela nous avons utilisé l'environnement de développement intégré Borland Jbuilder 8. Premièrement, nous avons conçu notre propre syntaxe qui permet de créer les entrées (des fichiers texte) de l'application : les TBox et les ABox.

En suite nous allons détailler les algorithmes développés, en premier lieu l'algorithme structurelle de teste de subsomption en utilisant la technique unfolding et en essayant de l'optimiser, puis nous allons voir la faiblesse de cette algorithme, de cet fait passant a l'algorithme des tableaux sémantiques.

La conclusion générale de ce mémoire propose un bilan et une analyse critique des travaux menés et expose les perspectives qu'ils ouvrent.

Chapitre I

Les ontologies comme outils de représentation de connaissances

1. Introduction:

Nées des besoins de représentation des connaissances, les ontologies sont à l'heure actuelle au cœur des travaux menés en Ingénierie des Connaissances (IC). Visant à établir des représentations à travers lesquelles les machines puissent manipuler la sémantique des informations. La construction des ontologies demande à la fois une étude des connaissances humaines et la définition de langages de représentation, ainsi que la réalisation de systèmes pour les manipuler. Les ontologies participent donc pleinement aux dimensions scientifique et technique de l'Intelligence Artificielle (IA) dite: scientifique : comme étude des connaissances humaines et plus largement de l'esprit humain, ce qui rattache l'IA aux sciences humaines, et technique comme création d'artefacts possédant certaines propriétés et capacités en vue d'un certain usage.

Les ontologies apparaissent ainsi comme des composants logiciels s'insérant dans les systèmes d'informations et leur apportant une dimension sémantique qui leur faisait défaut jusqu'ici. Le champ d'application des ontologies ne cesse de s'élargir et couvre les systèmes conseillers, les systèmes de résolution de problèmes ou les systèmes de gestion de connaissances. Un des plus grands projets basés sur l'utilisation d'ontologies consiste à ajouter au Web une véritable couche de connaissances permettant, dans un premier temps, des recherches d'informations au niveau sémantique et non plus simplement syntaxique. A terme, il est prévu que des applications internet pourront mener des raisonnements utilisant les connaissances stockées sur la Toile.

L'enjeu de l'effort engagé est de rendre les machines suffisamment sophistiquées

pour qu'elles puissent intégrer le sens des informations, qu'à l'heure actuelle, elles ne font que manipuler formellement. Mais en attendant, de nombreux problèmes théoriques et pratiques restent à résoudre.

Ce chapitre a pour but de présenter sans exhaustivité l'état de l'art en ingénierie ontologique. En explicitant la notion d'ontologie à travers les besoins auxquels elle répond, puis en décrivant la place qu'occupent les ontologies dans le processus de représentation des connaissances. En fin, en présentant ce qui signifie les connaissances inférentielles.

2 La notion d'ontologie:

Introduit en Intelligence Artificielle (IA) il y a 20 ans, le terme d'ontologie est cependant usité en philosophie depuis le XIXième siècle. Dans ce domaine, l'Ontologie désigne l'étude de ce qui existe, c'est à dire l'ensemble des connaissances que l'on a sur le monde **[WEL 01].** En IA, de façon moins ambitieuse, on ne considère que des ontologies, relatives aux différents domaines de connaissances. C'est à l'occasion de l'émergence de l'Ingénierie des Connaissances que les ontologies sont apparues en IA, comme réponses aux problématiques de représentation et de manipulation des connaissances au sein des systèmes informatiques.

2.1 L'origine des ontologies :

L'Ingénierie des Connaissances (IC) est une branche de l'IA issue de l'étude des Systèmes Experts (SE). Si ces derniers n'avaient pour objet que la résolution automatique de problèmes, les **Systèmes à Base de Connaissances (SBC)** qui leur ont succédé sont censés permettre le stockage et la consultation de connaissances, le raisonnement automatique sur les connaissances stockées, la modification des connaissances stockées, et, avec le développement des réseaux, le partage de connaissances entre systèmes informatiques. De manière générale, il ne s'agit plus de faire manipuler en aveugle des connaissances à la machine, qui restitue à la fin la solution du problème, mais de permettre un dialogue, une coopération entre le système et l'utilisateur humain (systèmes d'aide à la décision, systèmes d'enseignement assisté par ordinateur, recherche d'information sur le web).

Le système doit donc avoir accès non seulement aux termes utilisés par l'être humain, mais également à la sémantique que ce dernier associe aux différents termes. Plus

précisément, les représentations symboliques utilisées dans les machines doivent avoir du sens aussi bien pour la machine que pour les utilisateurs, « avoir du sens » signifiant ici que l'on peut relier les informations représentées à d'autres informations. Pour cela, la représentation des connaissances sous forme de règles logiques, utilisée dans les Systèmes Experts, ne suffit plus. Pour modéliser la richesse sémantique des connaissances, de nouveaux formalismes sont introduits, qui représentent les connaissances au niveau *conceptuel*, y compris la « structure cognitive » d'un domaine .*« Most KR formalisms differ from pure first-order logic in their structuring power, i.e. their ability to make evident the structure of a domain »* **[GUA 94B].** Les langages à base de frames, **les logiques de descriptions** et les graphes conceptuels sont des exemples de tels formalismes. Ces langages permettent de représenter les concepts sous-jacents à un domaine de connaissance, les relations qui les lient, et la sémantique de ces relations, indépendamment de l'usage que l'on souhaite faire de ces connaissances. On découple alors la représentation des connaissances de celle des mécanismes inférentiels.

Modéliser des connaissances ne peut se faire que dans un domaine de connaissance donné, et pour un but donné, condition nécessaire à l'unicité de la sémantique associée aux termes du domaine. Certains auteurs estiment cependant que les ontologies sont, par nature, destinées à être réutilisées **[FER 97]**, et s'attachent à construire des ontologies dont la sémantique soit indépendante de tout objectif opérationnel.

De plus, il ne faut pas oublier que l'ordinateur ne manipule que des symboles, il n'est qu'un médiateur de connaissances, à l'instar d'un livre, et la sémantique des représentations symboliques manipulées n'est construite que par les utilisateurs de l'ordinateur **[BAC 99]**. Cette sémantique est cependant fortement contrainte par la façon dont les symboles sont utilisés dans la machine. C'est pourquoi N. GUARINO plaide dans **[GUA 94A]** pour l'introduction d'un niveau ontologique entre le niveau conceptuel (où l'interprétation est subjective, comme au niveau linguistique), et le niveau épistémologique, structuré (où l'interprétation est arbitraire). Au niveau ontologique, les primitives utilisées pour représenter les connaissances ne sont plus des mots du langage naturel, ou des primitives conceptuelles, et pas encore des prédicats logiques, mais des énoncés qui donnent le sens des connaissances, avec une interprétation contrainte.

Le niveau ontologique apparait donc comme un pont entre la sémantique interprétative au travers de laquelle l'utilisateur interprète les termes fournis par la machine et la sémantique opérationnelle au travers de laquelle la machine manipule les symboles **[DEC 00]**. Fortement liée à la sémantique interprétative, la sémantique de l'ontologie est, idéalement, indépendante de la sémantique opérationnelle, mais cependant partiellement contrainte par cette dernière. Les ontologies sont donc des représentations de connaissances, contenant des termes et des énoncés qui spécifient la sémantique d'un domaine de connaissance donné dans un cadre opérationnel donné. Le terme « ingénierie ontologique » a été proposé par R. MIZOGUCHI en 1997 pour désigner un nouveau champ de recherche ayant pour but la construction de systèmes informatiques tournés vers le contenu, et non plus vers les mécanismes de manipulation de l'information.

2.2 Les ontologies au sein du processus de représentation des connaissances :

A l'heure actuelle, un certain consensus s'est établi sur le rôle des ontologies dans ce processus, consensus bâti autour de la formule de T. GRUBER, *« une ontologie est une spécification explicite d'une conceptualisation »* **[GRU 93]**.

La construction d'une ontologie n'intervient donc qu'après que le travail de conceptualisation ait été mené à bien. Ce travail consiste à identifier, au sein d'un corpus, les connaissances spécifiques au domaine de connaissances à représenter. *« A conceptualisation is an abstract, simplified view of the world that we wish to represent for some purpose »* **[GRU 93]**. N. GUARINO affine la définition de T. GRUBER en considérant les ontologies comme des spécifications partielles et formelles d'une conceptualisation **[GUA 95].**

Les ontologies sont formelles car exprimées sous forme logique, Partielles car une conceptualisation ne peut pas toujours être entièrement formalisée. Aussi, parce que les formalismes opérationnels présentent une faible tolérance d'interprétation (J. NOBÉCOURT dans **[NOB 00]**): passer directement d'une ontologie informelle à une ontologie totalement formelle et non-ambiguë.

Il est donc nécessaire de pouvoir construire une première modélisation semi-formelle, partiellement cohérente, correspondant à une conceptualisation semi-formalisée. On parle alors d'*ontologie conceptuelle*, semi-formelle, et le processus de spécification en question est appelé *ontologisation* **[KAS 00].** En effet, une ontologie n'est pas

opérationnelle, au sens où elle n'incluse pas de mécanismes de raisonnement, puisqu'elle doit justement être indépendante de tout objectif opérationnel. Le langage cible doit donc permettre de représenter les différents types de connaissances (connaissances terminologiques, faits, règles et contraintes) et de manipuler ces connaissances à travers des mécanismes adaptés à l'objectif opérationnel du système conçu. Ce processus de traduction est appelé *opérationnalisation.*

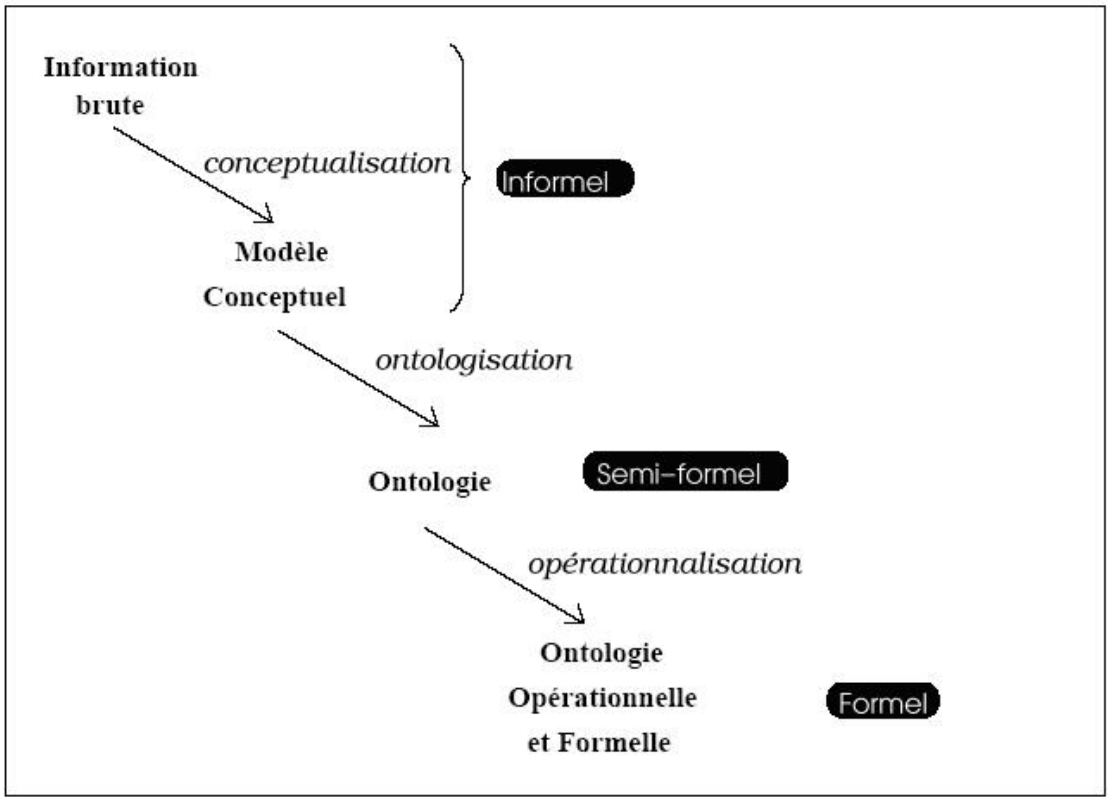

FIG 1- Construction d'une ontologie opérationnelle **[FUR 04].**

Le processus général de représentation des connaissances peut donc être découpé en 3 phases (cf. figure 1) :

La conceptualisation : identification des connaissances contenues dans un corpus représentatif du domaine. Ce travail doit être mené par un expert du domaine, assisté par un ingénieur de la connaissance ;

L'ontologisation : formalisation, autant que possible, du modèle conceptuel obtenu à l'étape précédente. Ce travail doit être mené par l'ingénieur de la connaissance, assisté de l'expert du domaine;

L'opérationnalisation : transcription de l'ontologie dans un langage formel et

opérationnel de représentation de connaissances. Ce travail doit être mené par l'ingénieur de la connaissance.

2.3 Les constituants d'une ontologie :

2.3.1 Connaissances et domaines de connaissance :

Une ontologie ne peut être construite que dans le cadre d'un domaine précis de la connaissance, ne serait-ce que parce que beaucoup de termes n'ont pas le même sens d'un domaine à l'autre, et qu'une sémantique non ambiguë doit être intégrée à l'ontologie. Certaines connaissances, qui peuvent constituer en elles-mêmes un domaine, sont utilisées dans tous les autres domaines. C'est le cas des notions générales liées à la causalité, au temps, à l'espace, etc. Savoir borner le domaine des connaissances à représenter demande donc une délimitation extrêmement précise de l'objectif opérationnel de l'ontologie.

D'autre part, dans le domaine de l'Ingénierie des Connaissances, le terme de connaissance a un sens forcément restreint : ne sont considérées que les connaissances (au sens large) susceptibles d'être formalisées, c'est-à-dire les connaissances peu ou prou techniques **[BAC 00]** : *« For knowledge-based systems, what exists is exactly that which can be represented »*. En outre, En manipulation automatique il n'y a connaissance que quand l'information présente dans la machine prend un sens pour l'utilisateur et ce sens doit être le même pour tous les utilisateurs. Les connaissances manipulées dans les SBC se doivent d'être des informations « actives », c'est-à-dire susceptibles d'influencer le déroulement de processus, de produire de nouvelles informations ou de permettre de prendre des décisions **[KAY 97]**. Seul ce type de connaissances « productives » parait offrir un intérêt pour l'automatisation. En conclusion, la construction d'une ontologie doit se faire à partir d'un champ de connaissances bien délimité par un objectif opérationnel clair, et portant sur des connaissances objectives dont la sémantique puisse être exprimée rigoureusement et formellement.

Une distinction est établie entre **les ontologies de domaine** portant sur des concepts renvoyant à des objets matériels ou à des concepts d'assez **bas niveau** (c'est-à-dire n'offrant que des possibilités limitées de raffinement) et les ontologies portant sur des concepts de **haut niveau** (**upper-ontologies**). Ces dernières décrivent des notions générales comme les notions d'objet, de propriété, d'état, de valeur, de moment, d'événement, d'action, de cause et d'effet **[SOW 00]**. Parmi les ontologies de haut

niveau, on trouve des ontologies qui vont décrire les notions utilisées dans toutes les ontologies pour spécifier les connaissances, telles que les sortes, les substances, les concepts, les relations, etc. Ces ontologies de représentation sont en fait indépendantes des différents domaines de connaissance, puisqu'elles décrivent des primitives cognitives communes aux différents domaines **[GUA 94B]**.

On considère également que les **processus de raisonnement appliqués aux connaissances** forment eux-mêmes un domaine de connaissance, que l'on peut représenter à l'aide d'une ontologie. En particulier, des ontologies ont été développées afin de représenter les connaissances génériques mises en œuvre lors de la résolution automatique de problèmes. Par exemple, B. CHANDRASEKARAN propose, dans **[CHA 98]**, une ontologie portant sur la résolution de problème par la décomposition en tâches et méthodes, ontologie indépendante des domaines d'application potentiels ; seules sont décrites les connaissances portant sur la façon d'utiliser d'autres connaissances, elles-mêmes non précisées. B. CHANDRASEKARAN considère d'ailleurs que, dans un système à base de connaissances utilisant des mécanismes inférentiels (c'est-à-dire un système non exclusivement destiné à la consultation de connaissances stockées), deux ontologies distinctes spécifiant les connaissances à manipuler et les connaissances de raisonnement doivent être intégrées.

Le domaine de connaissance sur lequel porte l'ontologie n'est d'ailleurs pas le seul critère permettant de la typer. Dans **[USC 96]**, M. USCHOLD considère que les ontologies varient suivant trois dimensions : le **degré de formalisme** de la représentation (qui varie continuèrent depuis le *rigoureusement formel* jusqu'au *hautement informel*), **l'objectif opérationnel** (communication entre utilisateurs, interopérabilité entre systèmes, application à un problème d'ingénierie comme la réutilisabilité de composants, résolution de problèmes) et **le sujet** (domaine de connaissance, connaissances de raisonnement, connaissances liées au modèle de représentation).

Une présentation plus fine des différents types d'ontologie présentés dans la figure 2 **[MIZ 97]**.

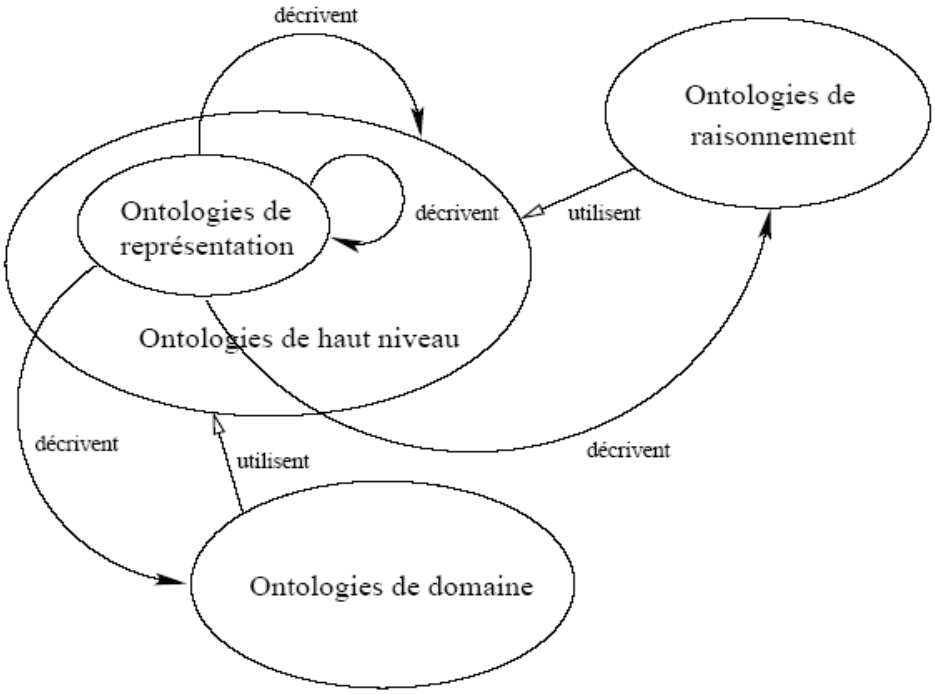

FIG. 2- Les différents types d'ontologies **[FUR 04].**

2.3.2 Les concepts et les relations :

Les connaissances portent sur des objets auxquels on se réfère à travers des concepts.
Un **concept** peut représenter un objet **matériel**, une **notion**, une **idée [USC 95]**.
Un concept peut être divisé en trois parties :

- **Un terme** (ou plusieurs),
- **Une notion** et
- **Un ensemble d'objets**.

• **Le terme** : identifie le concept.

• **La notion** : également appelée intension du concept, **contient la sémantique du concept**, exprimée en termes de propriétés et d'attributs, de règles et de contraintes.

• **L'ensemble d'objets** : également appelé extension du concept, regroupe les objets manipulés à travers le concept ; ces objets sont appelés instances du concept.

Par exemple, le terme « table » renvoie à la fois à la notion de table comme objet de type « meuble » possédant un plateau et des pieds, et à l'ensemble des objets de ce type.

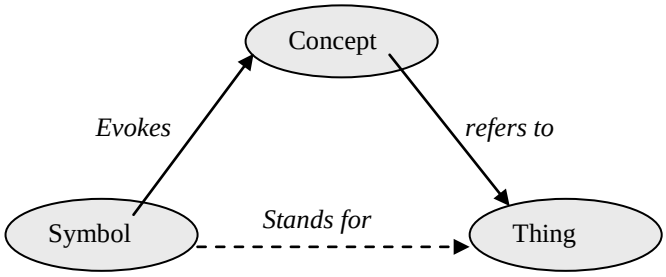

FIG. 3- *Le triangle du sens*

Les deux aspects d'un concept sont assez différents, en particulier par le fait que deux extensions peuvent ne pas être disjointes, alors que deux intensions s'excluent mutuellement par au moins une propriété. B. BACHIMONT utilise les termes :

Concept formel *:* pour désigner l'extension d'un concept.

Concept sémantique *:* pour désigner l'intension d'un concept **[BAC 00].**

Un concept est ainsi doté d'une **sémantique référentielle** (celle imposée par son **extension**) et d'une **sémantique différentielle** (celle imposée par son **intension**).

L'articulation entre les aspects référentiels et différentiels des concepts est **délicate**.

On estime que **l'intension** d'un concept permet à **elle seule** d'en préciser le **sens**.

Deux intentions s'excluent mutuellement par au moins une propriété.

Deux extensions peuvent ne pas être disjointes.

Un concept avoir une extension vide. C'est un concept générique.

Deux concepts peuvent partager la même extension.

Deux concepts peuvent partager le même terme et la même extension.

Un autre débat non tranché porte sur les **propriétés** contenues dans les intensions des concepts.

Propriété **essentielle** $\Leftrightarrow$ **(**la suppression de cette propriété c'est la disparition du concept en tant que tel**) [BOU 95]**.

Propriété **non essentielle** $\Leftrightarrow$ **(**propriétés vraies dans un contexte donné $\Rightarrow$ leur disparition ne modifie pas intrinsèquement le concept**)**.

N. GUARINO ne considère comme ontologique que les propriétés nécessaires **[GUA 95]**. G. KASSEL admet dans les ontologies des propriétés incidentes, c'est-à-dire vraies seulement dans le cadre applicatif **[KAS 02].** On peut cependant considérer que dans ce cadre, ces propriétés incidentes sont nécessaires à la complétude de la définition du concept, qui est plus spécifique que celui considéré dans le cadre général

(une définition est complète lorsqu'elle ne possède aucun contre-exemple).

L'exemple du concept de table, montre que cette notion ne peut se définir qu'en utilisant d'autres concepts comme « meuble », « plateau » et « pied ».

De fait, les concepts manipulés dans un domaine de connaissances sont organisés au sein d'un **réseau de concepts**. L'ensemble des concepts y est structuré hiérarchiquement et les concepts sont liés par des propriétés conceptuelles. La propriété utilisée pour structurer la hiérarchie des concepts est la **subsomption**.

(Si C2 est plus spécifique que C1) $\Rightarrow$C1 subsume C2.

Un concept C1 subsume un concept C2 si toute propriété sémantique de C1 est aussi une propriété sémantique de C2.

(C1 subsume C2) $\Rightarrow$ L'**extension** de **C2** est plus **réduite** que celle de **C1**.

(C1 subsume C2) $\Rightarrow$ L'**intention** de **C2** est plus **riche** que celle de **C1**.

Les propriétés portant sur un concept sont :

- la généricité : un concept est générique s'il n'admet pas d'extension. *e.g:* la vérité est un concept générique ;

- l'identité : (propriété proposée par N. GUARINO) un concept porte une propriété d'identité si cette propriété permet de conclure quant à l'identité de deux instances de ce concept. Cette propriété peut porter sur des attributs du concept ou sur d'autres concepts. *e.g:* le concept d'étudiant porte une propriété d'identité liée au numéro de l'étudiant, deux étudiants étant identiques s'ils ont le même numéro ;

- la rigidité : (propriété proposée par N. GUARINO) un concept est rigide si toute instance de ce concept en reste instance dans touts les mondes possibles. *e.g:* humain est un concept rigide, étudiant est un concept non rigide ;

- l'anti-rigidité : (propriété proposée par N. GUARINO) un concept est anti-rigide si toute instance de ce concept est essentiellement définie par son appartenance à l'extension d'un autre concept. *e.g*: étudiant est un concept anti-rigide car l'étudiant est avant tout un humain ;

- l'unité : (propriété proposée par N. GUARINO) un concept est un concept unité si, pour chacune de ses instances, les différentes parties de l'instance sont liées par une relation qui ne lie pas d'autres instances de concepts. *e.g*: les deux parties d'un couteau, manche et lame sont liées par une relation « emmanché » qui ne lie que cette lame et ce manche.

Les propriétés portant sur deux concepts sont :

- l'équivalence : deux concepts sont équivalents s'ils ont même extension. *e.g:* étoile du matin et étoile du soir ;

- la disjonction : (on parle aussi d'incompatibilité) deux concepts sont disjoints si leurs extensions sont disjointes. *e.g:* homme et femme ;

- la dépendance : (propriété proposée par GUARINO) un concept C1 est dépendant d'un concept C2 si pour toute instance de C1 il existe une instance de C2 qui ne soit ni partie ni constituant de l'instance de C2. *e.g:* parent est un concept dépendant de enfant (et vice-versa).

Les propriétés exposées ici forment elles-mêmes une **taxinomie** que l'on peut hiérarchiser **[KAS 02]**. De plus, certaines propriétés peuvent s'appliquer aux propriétés elles-mêmes **[GUA 00]**. C. WELTY considère d'ailleurs les concepts et les relations comme des propriétés, indistinctement **[WEL 01]**. On définit alors les propriétés précédentes comme des **méta-propriétés** portant sur n'importe quelle propriété.

N. GUARINO et C. WELTY définissent, à l'aide de ces méta-propriétés, des types de concepts qui regroupent chacun les concepts vérifiant certaines propriétés. Ces types de concepts sont :

- <u>les sortes</u> : qui se déclinent en

Types **:** rigides et portant leur propre identité, comme « personne », « chat », « eau » ;

Quasi-types **:** anti-rigides et portant leur propre identité, comme « herbivore » ;

Modèles matériels **:** anti-rigides et dépendant, comme « étudiant » et « nourriture » ;

Phases **:** portant leurs propres identités, anti-rigides et indépendantes comme « papillon »

Mixtes **:** portant leurs propres identités, non rigides, comme « mâle ».

- <u>les non sortes</u> qui se déclinent en

Catégories **:** rigides, ne portant pas d'identité, comme « chose », « lieu » ;

Modèles formels **:** anti-rigides et dépendants, comme « acteur », « patient » ;

Attributions **:** non rigides et non sortes, comme « rouges ».

Il est également possible de classer les propriétés que l'on utilise à l'aide d'autres critères. C. WELTY propose ainsi de distinguer les **propriétés intrinsèques**, qui ne sont liées qu'au concept lui-même, comme la généricité et les **propriétés extrinsèques**, qui font intervenir d'autres concepts dans leur définition **[WEL 01]**. Des modalités peuvent également être introduites au niveau des propriétés qui ne sont

pas de nature modale, comme l'identité. Un concept pourrait alors porter possiblement (i.e. dans certains mondes) ou nécessairement (i.e. dans tous les mondes possibles) un critère d'identité.

En plus des propriétés, l'intension d'un concept peut contenir des **attributs**. Un attribut peut être une instance de concept. D'autre part, on peut avoir besoin d'affecter des attributs (concepts ou instances) à une instance d'un concept **[GOM 96]**. Par exemple, une Ferrari, instance d'« automobile », porte un attribut « couleur rouge » instance du concept « couleur ».

Si certains liens conceptuels existant entre les concepts peuvent s'exprimer à l'aide de propriétés portées par les concepts, d'autres doivent être représentés à l'aide de **relations** autonomes. Une relation permet de **lier** des instances de concepts, ou des concepts génériques.

Une **Relation** est caractérisée par :

un **terme** (voire plusieurs) et

une **signature** qui précise le nombre d'instances de concepts que la relation lie,

leurs **types** et **l'ordre** des concepts, c'est–à–dire la façon dont la relation doit être lue.

La relation « écrit » lie une instance du concept « personne » et une instance du concept « texte », dans cet ordre.

Trancher entre la représentation d'une notion sous forme d'un concept ou d'une relation peut parfois s'avérer nécessaire. Par exemple, l'écriture d'un texte peut être vue comme un concept et on exprimera le fait qu'une personne écrit en disant qu'elle est en relation de type « mener une action », l'action étant elle-même l'écriture qui sera en relation avec un « texte ». Le **choix** dépend alors essentiellement **de l'objectif opérationnel de l'ontologie** et du **contexte d'utilisation** de la notion. De même, il convient de choisir entre l'utilisation de relations ou d'attributs pour représenter les liens entre concepts.

Tout comme les concepts, les relations peuvent être spécifiées par des propriétés dont une liste, non exhaustive, est donnée ci-après. Les relations sont organisées de manière **hiérarchisée** à l'aide de la propriété de **subsomption** décrite précédemment. Les remarques faites au sujet de l'organisation des propriétés s'appliquent également dans ce cas.

Les propriétés intrinsèques à une relation sont :

- les propriétés algébriques : symétrie, réflexivité, transitivité ;

- la cardinalité : nombre possible de relations de ce type entre les mêmes concepts (ou instances de concept).

Les relations portant une cardinalité représentent souvent des attributs. *e.g:* une pièce a au moins une porte, un humain a entre zéro et deux jambes.

Les propriétés liant deux relations sont :

- l'incompatibilité : deux relations sont incompatibles si elles ne peuvent lier les mêmes instances de concepts. *e.g:* les relations « être rouge » et « être vert » sont incompatibles ;

- l'inverse : deux relations binaires sont inverses l'une de l'autre si, quand l'une lie deux instances I1 et I2, l'autre lie I2 et I1. *e.g:* les relations « a pour père » et « a pour enfant » sont inverses l'une de l'autre ;

- l'exclusivité : deux relations sont exclusives si, quand l'une lie des instances de concepts, l'autre ne lient pas ces instances, et vice-versa. L'exclusivité entraine l'incompatibilité. *e.g:* l'appartenance et le non appartenance sont exclusives.

Les propriétés liant une relation et des concepts sont :

- le lien relationnel : (propriété proposée par G. KASSEL) il existe un lien relationnel entre une relation R et deux concepts C1 et C2 si, pour tout couple d'instances des concepts C1 et C2, il existe une relation de type R qui lie les deux instances de C1 et C2. Un lien relationnel peut en outre être contraint par une propriété de cardinalité, ou porter directement sur une instance de concept **[KAS 02]**. *e.g.* : il existe un lien relationnel entre les concepts « texte » et « auteur » d'une part et la relation « a pour auteur » d'autre part.

- la restriction de relation : (propriété proposée par G. KASSEL) pour tout concept de type C1, et toute relation de type R liant C1, les autres concepts liés par la relation sont d'un type imposé. *e.g:* si la relation « mange » portant sur une « personne » et un « aliment » lie une instance de « végétarien », concept subsumé par « personne », l'instance de « aliment » est forcément instance de « végétaux ».

3. Les connaissances inférentielles :

Décrire les connaissances en termes de concepts, de relations entre ces concepts et de propriétés sur ces concepts et relations ne suffit généralement pas pour atteindre l'objectif opérationnel d'un Système à Base de Connaissances. *« De façon un peu*

caricaturale, on pourrait dire que ce ne sont pas les connaissances en elles-mêmes qui sont intéressantes. En fait, ce qui doit être le centre d'intérêt pour le modélisateur, c'est davantage leur mise en œuvre, leur concrétisation dans une action pour atteindre un but car c'est bien l'activité qu'on cherche à assister et non les connaissances en elles-mêmes » **[TEU 01]**. Il s'agit également de tirer au maximum parti de ce qui fait la spécificité du support informatique par rapport au support écrit traditionnel, c'est-à-dire son aspect dynamique.

Comparé à l'oral qui n'offre pas de support, et à l'écrit traditionnel, un système informatique peut manipuler les connaissances pour en inférer de nouvelles **[BAC 99]**.

Ces connaissances opérationnelles peuvent être des **faits**, des **règles**, ou des **contraintes**. Un fait est un énoncé vrai, non implicatif, un axiome qui participe à la description du monde cognitif dans lequel s'inscrit le SBC. L'énoncé « l'entreprise E compte 20 salariés » est un exemple de fait. Une règle permet d'inférer de nouvelles connaissances et contient donc une implication. La règle « si une entreprise compte X salariés, alors elle paye X*100 euros de charges » permet de calculer les charges d'une entreprise. Les contraintes permettent de spécifier des impossibilités ou des obligations. Par exemple, on peut vouloir spécifier que toute société importante possède obligatoirement un conseil d'administration et qu'une société ne peut avoir qu'un seul PDG. Le choix entre l'utilisation d'une règle ou d'une contrainte pour représenter une connaissance n'est pas toujours facile, et dépend encore une fois de la façon dont les connaissances vont être utilisées au niveau opérationnel.

De plus, certains auteurs considèrent que ces types de connaissances doivent être intégrés dans les propriétés des concepts, des relations ou des instances de concepts. En effet, généralement, un fait apparait comme une propriété d'une instance de concept. Une règle apparait plutôt comme une propriété d'un concept ou d'une relation car elle ne porte *a priori* pas que sur des instances. Par exemple, une règle peut servir à déterminer la valeur d'un attribut d'un concept **[GOM 96]**. Les contraintes, quant à elles, peuvent porter aussi bien sur des concepts ou relations que sur des instances de concepts. Pour S. STAAB et A. MAEDCHE, les axiomes sont à intégrer dans les ontologies comme des objets liés aux concepts et relations, de manière à apparaître comme des propriétés **[STA 00]**.

Ces choix de représentation des connaissances de raisonnement vont se doubler d'un

choix portant sur le mode d'utilisation de ces connaissances. En effet, lorsqu'on manipule des connaissances, une bonne partie de nos réflexions sont implicites, en particulier quand elles sont à tel point évidentes que les exprimer apparaitrait ridicule. Les connaissances inférentielles vont donc devoir être classées en connaissances implicites, c'est-à-dire utilisées par le système mais non exprimées à l'intention des utilisateurs, et les connaissances explicites qui doivent être exprimées. Ce choix est lié au contexte opérationnel de l'ontologie.

4. Conclusion :

Conçues comme réponse aux problèmes posés par l'intégration de connaissances au sein des systèmes informatiques, les ontologies apparaissent désormais comme une clé pour la manipulation automatique de l'information au niveau sémantique.

Les principaux projets utilisant des ontologies ne visent que la gestion de connaissances au niveau sémantique. Les ontologies pourraient permettre à terme la création de systèmes capables non seulement de gérer des connaissances mais aussi de raisonner sur ces connaissances et, pourquoi pas, d'en produire de nouvelles.

La définition d'une méthodologie unifiée de construction et de validation des ontologies est nécessaire. Cette unification doit porter sur les principes de structuration sémantique des connaissances mais également sur les langages opérationnels de représentation. Ces langages doivent d'autre part être enrichis pour permettre l'expression et la manipulation de connaissances inférentielles. Parmi ces langages on trouve les logiques de description qui, a la différence des autres, ils sont équipés avec une sémantique basée sur une logique formelle, et possède des mécanismes de raisonnement solide.

Chapitre II

Les logiques de descriptions

1 Introduction:

Les logiques de descriptions est le nom le plus récent d'une famille d'un formalisme de représentation de connaissance qui représente la connaissance d'un domaine d'application par définition des concepts pertinents du domaine (sa terminologie), et ensuite utiliser ces concepts pour spécifier les propriétés des objets et des individus qui se trouvent dans le domaine.

Comme le nom « logiques de descriptions » indique, un des caractéristiques de ces logiques est que, a la différence d'un certain de ces prédécesseurs, ils sont équipés avec une sémantique basée sur une logique formelle. Autre caractéristique distinguée est que les DL mettent l'accent sur le raisonnement comme un service central.

Le raisonnement permet d'inférer des connaissances représentées implicitement à partir des autres contenues explicites dans les bases de connaissances.

Les logiques de description supportent des modèles d'inférence se trouvent dans plusieurs applications des systèmes de traitement d'information intelligent, et utilisés par l'humain pour structurer et comprendre le monde : classification des concepts et des individus.

La classification des concepts détermine la relation subconcept-superconcept (appelée relation de subsomption dans les logiques de description) entre les concepts d'une terminologie, et ça permet de structurer la terminologie dans une forme d'hiérarchie de subsomption. Cette hiérarchie fournie l'information utile pour la connexion entre différentes concepts, et elle peut être utilisé pour accélérer les services d'inférences.

La classification des individus détermine si un individu est une instance d'un certain concept (cette relation d'instanciation est impliquée par la description d'individu et la définition d'un concept) et ça peut fournir des informations utiles concernant les

propriétés d'individu.

Les logiques de descriptions sont les descendants des « réseaux d'héritage structurés » **[BRA 77],** qui est introduit pour surmonter l'ambiguïté de ses prédécesseurs « réseaux sémantiques » et « frames » et qui ont réalisé pour la première fois dans le système KL-One **[BRA 85]**.

Les trois idées suivantes exprimées dans le travail de Brachman sur les réseaux d'héritage structurés ont largement façonné le développement suivant des logiques de description :

- les blocks basiques de construction syntactique sont les concepts atomiques (prédicats du premier ordre) les rôles atomiques (prédicats binaires) et les individus (les constantes).
- Le pouvoir expressif de langages est restreint tel qu'il utilise un petit ensemble (épistémologiquement adéquat) de constructeurs pour construire des concepts et des rôles complexes.
- Les connaissances implicites autour de concepts et individus peuvent être inférés automatiquement avec l'aide des procédures d'inférence. En particulier la relation de subsomption entre concepts et la relation d'instanciation entre individus et concepts joue un rôle important : à la différence du lien IS-A dans les réseaux sémantiques qui sont explicitement introduits par l'utilisateur, la subsomption et l'instanciation sont inférés depuis la définition des concepts et les propriétés des individus.

Après la première sémantique basée sur la logique du système KL - ONE comme langages de RC proposés, les procédures d'inférence comme la subsomption doivent être fournies avec un sens précis, qui a conduit à des premières investigations formelles des propriétés de calcul de ces langages. Il s'est avéré que les langages utilisées dans les systèmes DL au début étaient trop expressives, qui a conduit à l'indécidabilité du problème de subsomption **[SCH 89; PAT 89]**. Les premiers résultats de complexités de raisonnement des pire des cas (worst-case complexity) s'avèrent irréductible (dans le sens d'être non-polynomial) ne s'oppose pas à une logique de description d'être utile dans la pratique, à condition que les techniques d'optimisation sophistiquées sont utilisés lors de la mise en œuvre d'un système fondé sur de telles logiques de descriptions.

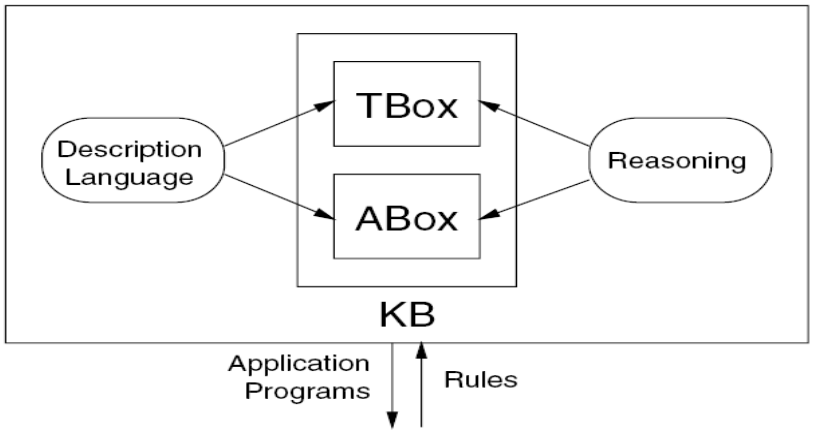

FIG. 4. Architecture des systèmes de représentation de connaissances basés sur les logiques de descriptions [FRA 03].

Ce chapitre introduit les éléments de base de la théorie des logiques de descriptions, en présentant les familles des langages de descriptions. Ensuite nous présenteront en détails les deux composants d'une base de connaissances basée sur les logiques de descriptions : la TBox et la ABox.

2. Définition du formalisme de base:

Une base de connaissances (KB pour Knowledge Representation) basée sur la logique de description comprend deux composants, la TBox et la ABox. La TBox introduit la terminologie, i.e., le vocabulaire du domaine d'une application, alors que la ABox contient des assertions quant a des individus nommés en termes de ce vocabulaire.

Le vocabulaire consiste en des concepts, qui dénotent des ensembles d'individus, et des rôles qui dénotent des relations binaires entre des individus. En plus des concepts atomiques et des rôles atomiques (les noms des concepts et des rôles), tous les systèmes DL permettent de construire des descriptions de concepts et de rôles plus complexes. Le langage de construction de descriptions constitue une caractéristique du système en question.

Les systèmes DL non seulement stockent la terminologie et les assertions, mais aussi offrent des services pour raisonner sur eux.

Les taches typiques de raisonnement sur la terminologie sont:

- Déterminer si la description est satisfiable (non contradictoire).
- Si une description est plus générale qu'une autre, en d'autre terme si la première subsume la deuxième.
- Le problème important pour la Abox est de trouver si l'ensemble d'assertions

est consistent C.à.d. s'il y a un modèle, ou si un individu est une instance d'un concept donné.

La satisfiabilité pour les descriptions et la consistance pour les assertions utilisées pour vérifier si la base est sémantiquement correcte.

Avec la subsomption on peut organiser les concepts de la terminologie dans une hiérarchie selon leurs généralités.

Dans n'importe quelle application, un système de RC est intégré dans un environnement plus vaste, Autres composants interagissent avec lui en interrogeant la base de connaissances et en la modifiant, et ça, en ajoutant et en retirant les concepts, les rôles, et les assertions.

2.1 Les langages de description :

Les descriptions élémentaires sont les concepts atomiques et les rôles atomiques à partir desquels des descriptions complexes peuvent être générées via les constructeurs de concepts et les constructeurs de rôles. Dans ce qui suit, on utilisera les lettres *A* et *B* pour designer des concepts atomiques, la lettre *R* pour un rôle atomique et les lettres *C* et *D* pour les descriptions de concepts.

Les langages de description sont distingués par rapport aux constructeurs qu'ils fournissent. Le langage *AL* (pour Attributive Langage) a été introduit dans **[SCH 91]** comme un langage minimal ayant un intérêt pratique. Les autres langages de la famille *AL* en sont une extension.

2.1.1 Le langage de description basique AL :

Les descriptions de concepts dans le langage *AL* sont générées selon la règle syntaxique suivante :

C, D →	A |	(concept atomique)
	⊤ |	(concept universel)
	⊥ |	(concept bottom)
	¬ A |	(négation atomique)
	C ⊓ D |	(intersection)
	∀R.C |	(restriction de valeur)
	∃R.⊤ |	(quantification existentiel limité)

Il est à noter, que dans le langage *AL*, la négation n'est appliquée qu'aux concepts atomiques et seulement le concept universel est permis au porté du quantificateur existentiel. D'autre part, le sous langage de *AL* obtenu en supprimant la négation

atomique est appelé FL^- alors que le sous langage de FL^- obtenu en supprimant le quantificateur existentiel limité est appelé FL_0 [FRA 03].

Pour donner des exemples de ce qui peut être exprimée en AL *Soient* Person *et* Femelle *deux concepts atomiques. Donc* Person ⊓ Femelle *et* Person ⊓¬Femelle *sont des concepts d´écrivant intuitivement les personnes qui sont femelle et les personnes qui ne sont pas femelle. En outre, étant donné un rôle atomique* enfant, *on peut construire les concepts* Person ⊓∃ enfant.⊤ *et* Person ⊓∀ enfant.Femelle *dénotant les personnes qui ont un enfant et les personnes dont tous les enfants sont une femelle. Enfin, en utilisant le concept bottom, on peut également d´écrire les personnes qui n'ont pas d'enfants par* Person ⊓∀ enfant. ⊥ .

En vue de définir une sémantique formelle des concepts AL, on considère une interprétation I définie par la donnée d'un ensemble non vide Δ^I (le domaine d'interprétation) et d'une fonction d'interprétation, qui assigne à chaque concept atomique A un ensemble $A^I \subseteq \Delta^I$ et à chaque rôle atomique R une relation binaire $R^I \subseteq \Delta^I \times \Delta^I$. Une fonction d'interprétation est étendue aux descriptions de concepts via la définition inductive suivante :

$$
\begin{aligned}
T^I &= \Delta^I \\
\bot^I &= \emptyset \\
(\neg A)^I &= \Delta^I - A^I \\
(C \sqcap D)^I &= C^I \cap D^I \\
(\forall R.C)^I &= \{a \in \Delta^I \mid \forall b, (a, b) \in R^I \rightarrow b \in C^I\} \\
(\exists R.T)^I &= \{a \in \Delta^I \mid \exists b, (a, b) \in R^I\}
\end{aligned}
$$

Deux concepts C et D sont dits équivalents, $C \equiv D$ si et seulement si $C^I = D^I$ pour toute interprétation I. Par exemple, on peut aisément vérifier que les concepts ∀enfant.Femelle ⊓∀enfant.Etudiant et ∀enfant.(femelle ⊓ Etudiant) sont équivalents.

2.1.2 La famille des langages AL :

D'autres langages, plus expressifs, peuvent être définis en rajoutant d'autres constructeurs au langage AL a savoir:

L'union de concepts, désignée par la lettre U, notée par C ⊔ D et interprétée par: $(C \sqcup D)^I = C^I \cup D^I$.

La quantification existentielle complète, désignée par la lettre $\mathcal{E}$, notée par ∃R.C et interprétée par:

$(\exists R.C)^I = \{a \in \Delta^I \mid \exists b, (a, b) \in R^I \wedge b \in C^I\}$

Les restrictions de nombres, désignées par la lettre *N* et notées par $\geq nR$ (restriction au moins) et par $\leq nR$ (restriction au plus) ou n représente un entier positif. Ces restrictions de valeurs sont interprétées respectivement comme suit:

$$(\geq nR)^I = \{a \in \Delta^I : \mid \{b \mid (a, b) \in R^I\} \mid \geq n\}$$

Et

$$(\leq nR)^I = \{a \in \Delta^I : \mid \{b \mid (a,b) \in R^I\} \mid \leq n\}$$

Où ' | . | ' dénote le cardinal d'un ensemble donné.

La négation de concepts arbitraires, désignée par la lettre (***C***), notée par ¬ C et interprétée de la façon suivante :

$$(\neg C)^I = \Delta^I - (C)^I$$

Avec ces nouveaux constructeurs, on peut par exemple décrire les personnes ayant soit pas plus un enfant soit au moins trois enfants dont un est une femelle comme suit:
Person ⊓ (≤ 1 hasChild ⊔ (≥ 3 has Child ⊓ ∃hasChild.Female)
Etendre le langage AL par n'importe quel sous ensemble des constructeurs précédents génère un langage AL particulier désigné par une chaîne de caractères de la forme :

AL [***U***][$\mathcal{E}$][***N***][***C***]

Ou une lettre dans l'appellation reflète la présence du constructeur correspondant. Par exemple, AL [$\mathcal{E}$][***N***] est l'extension de AL par la quantification existentielle complète et les restrictions de nombres.
D'un point de vue sémantique, on a C ⊔ D≡¬(¬C ⊓ ¬D) et ∃R.C≡¬∀R.¬C
Par conséquent, l'union et la quantification universelle complète peuvent être exprimées via la négation et vice versa. Ainsi, on utilisera la lettre ***C*** au lieu des lettres ***U*** $\mathcal{E}$ dans les noms de langages. Par exemple, on écrira AL***CN*** au lieu de AL***U*** $\mathcal{E}$ ***N***.

2.1.3 Les langages de description et la logique des prédicats du premier ordre :

La sémantique des concepts identifie les logiques de descriptions comme étant des fragments de la logique des prédicats du premier ordre. En fait, puisqu'une

interprétation *I* assigne à chaque concept atomique et à chaque rôle une relation unaire et une relation binaire sur Δ^I respectivement, les concepts et les rôles peuvent être vus comme étant des prédicats unaires et binaires **[FRA 03].** Donc, n'importe quel concept C peut être traduit par une formule logique $\Phi C(x)$ avec une seule variable libre x tel que pour toute interprétation *I*, l'ensemble des éléments de Δ^I qui satisfont $\Phi C(x)$ est exactement C^I :

_Un concept atomique *A* est traduit par la formule $A(x)$;

– Les constructeurs intersection, union et négation sont traduits par la conjonction logique, la disjonction et la négation respectivement ;

– si $\Phi C(x)$ est la traduction de *C* et *R* est un rôle atomique, alors la restriction de valeurs et la quantification universelle sont capturées par les formules :

$$\Phi_{\exists R.C(y)} = \exists x.R(y, x) \wedge \Phi_{C(x)}$$

$$\Phi_{\forall R.C(y)} = \forall x.R(y, x) \rightarrow \Phi_{C(x)}$$

ou y est une nouvelle variable ;

– Enfin, les restrictions de nombres sont exprimées par les formules :

$$\Phi_{\geq nR}(x) = \exists y_1, \ldots, y_n.R(x, y_1) \wedge \ldots R(x, y_n) \wedge \bigwedge_{i<j} y_i \neq y_j$$

$$\Phi_{\leq nR}(x) = \forall y_1, \ldots, y_{n+1}.R(x, y_1) \wedge \ldots R(x, y_{n+1}) \rightarrow \bigvee_{i<j} y_i = y_j$$

Etant donné que les concepts soient traductibles en logique des prédicats, on peut dire qu'une syntaxe spéciale est dénuée d'intérêts. Néanmoins, les translations précédentes montrent, en particulier pour la restriction de nombres, que la syntaxe à variable free des logiques de description est vraiment plus concise. Ceci facilite également le développement d'algorithmes.

2.2 Terminologies :

Nous avons vu comment nous pouvons former des descriptions complexes des concepts pour décrire les classes des objets. Maintenant, nous introduisons les axiomes terminologiques, qui font des déclarations sur la manière dont les concepts ou les rôles sont liés les uns aux autres. Ensuite, nous distinguons les définitions comme des axiomes spécifiques et d'identifier les terminologies comme des ensembles de définitions qui nous permet d'introduire des concepts atomiques comme

des abréviations ou des noms des concepts complexes. Si les définitions dans une terminologie contiennent cycles, nous pourrions avoir à adopter la sémantique de point fixe afin de les rendre sans équivoque.

2.2.1 Axiomes terminologiques :

En général, les axiomes terminologiques sont de la forme

$$C \sqsubseteq D \ (R \sqsubseteq S)$$

Ou

$$C \equiv D \ (R \equiv S)$$

Où C, D sont des concepts (et R, S son des rôles). Les axiomes du premier type sont appelés inclusions alors que ceux du second type sont dits égalités.

Pour simplifier, nous traiterons par la suite uniquement les axiomes liés aux concepts.

Sémantiquement, une interprétation ***I*** satisfait une inclusion $C \sqsubseteq D$ si et seulement si $C^I \subseteq D^I$ et satisfait une égalité $C \equiv D$ si et seulement si $C^I = D^I$.

Si Σ est un ensemble d'axiomes, alors ***I*** satisfait Σ si et seulement Si ***I*** satisfait chaque élément de Σ. Si ***I*** satisfait un axiome (resp. un ensemble d'axiomes), alors ***I*** est dite modèle de cet axiome (resp. l'ensemble d'axiomes). Deux axiomes ou deux ensembles d'axiomes sont équivalents si et seulement s'ils ont les mêmes modèles.

2.2.2 Définitions :

Une égalité dont le membre gauche est un concept atomique est une définition.

Les définitions sont utilisées pour affecter des noms symboliques à des descriptions complexes. Par exemple, par l'axiome:

$$\text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild.Person}$$

On associe à la description Woman $\sqcap \exists$hasChild.Person le nom Mother.

Les noms symboliques peuvent être utilisés comme des abréviations dans d'autres descriptions. Si, par exemple, on a défini le concept Father par analogie au concept Mother on peut définir le concept Parent comme suit:

$$\text{Parent} \equiv \text{Mother} \sqcup \text{Father}.$$

On appelle un ensemble de définitions Σ une terminologie ou une TBox si aucun nom symbolique n'est défini plus d'une fois, i.e. pour chaque concept A, il existe au plus un axiome dans Σ dont le membre gauche est A.

$$\text{Woman} \equiv \text{Person} \sqcap \text{Female}$$

$$\text{Man} \equiv \text{Person} \sqcap \neg \text{Woman}$$

$$\text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild.Person}$$

$$\text{Father} \equiv \text{Man} \sqcap \exists \text{hasChild.Person}$$

$$\text{Parent} \equiv \text{Mother} \sqcup \text{Father}$$

$$\text{GrandMother} \equiv \text{Mother} \sqcap \exists \text{haschild.Parent}$$

$$\text{MotherWithManyChildren} \equiv \text{Mother} \sqcap \geq 3\text{hasChild}$$

$$\text{MotherWithoutDaughter} \equiv \text{Mother} \sqcap \forall \text{hasChil.} \neg \text{Woman}$$

$$\text{Wife} \equiv \text{Woman} \sqcap \exists \text{hasHusband.Man}$$

FIG. 5. La terminologie (TBox) des concepts liés à des relations familiales.

2.2.3 Terminologies définitoriales :

Etant donnée une terminologie T. On divise les concepts atomiques y figurant en deux ensembles, les symboles de nom ou les symboles définis T_N qui figurent dans la partie gauche d'un certain axiome et les symboles de base ou les symboles primitifs T_B qui figurent uniquement dans les parties droites des axiomes. Normalement, une terminologie définit les symboles de nom en termes des symboles de base.

Une interprétation de base pour T est une interprétation qui interprète uniquement les symboles de base. Soit *J* une telle interprétation. Une interprétation *I* qui interprète également les symboles de nom est une extension de *J* si elle a le même domaine que *J*, i.e, $\Delta^J = \Delta^I$, et interprète les symboles de base de la même manière que *J*.

On dit que T est définitoriale si chaque interprétation de base admet exactement une extension qui est un modèle de T. Autrement dit, si on connaître l'interprétation des symboles de base, et T est définitoriale, alors le sens des concepts de nom est complètement défini.

2.2.4 Terminologies acycliques et terminologies cycliques :

Voir si une terminologie est définitoriale ou non est lié à voir si ses définitions contiennent des cycles ou non. Par exemple, la définition

$$\text{Human} \equiv \text{Animal} \sqcap \forall \text{hasParent.Human}$$

contient un cycle. En général, les cycles dans une terminologie T sont définis comme suit. Soient *A* et *B* deux concepts atomiques dans T. On dit que *A* utilise directement *B*

si *B* apparaît dans le membre droit de la définition de *A*, et on appelle 'utilise' la fermeture transitive de la relation 'utilise directement'.

Donc une terminologie T contient un cycle si et seulement s'il existe un concept atomique dans T qui utilise lui même. Autrement, T est dite acyclique.

Si T est acyclique, alors elle est définitoriale. La raison est qu'on peut effectuer itérativement l'expansion des définitions de T en remplaçant chaque occurrence d'un symbole de nom dans la partie droite d'une définition par les concepts qu'il représente. Comme il n'a a pas de cycles, le processus se termine en générant une terminologie T' contenant des définitions de la forme $A \equiv C'$, ou C' contient uniquement des symboles de base et non pas des symboles de nom. T' est dite expansion de T et sa taille peut être exponentielle par rapport a celle de T et elle lui est équivalente. La TBox indiqué dans la figure 5 est acyclique, par conséquent, nous pouvons calculer l'expansion, ce qui est indiqué dans la Figure 6.

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬(Person ⊓ Female)

Mother ≡ (Person ⊓ Female) ⊓ ∃hasChild.Person

Father ≡ (Person ⊓ ¬(Person ⊓ Female)) ⊓ ∃hasChild.Person

Parent ≡ ((Person ⊓ ¬(Person ⊓ Female)) ⊓ ∃hasChild.Person) ⊔ ((Person ⊓ Female) ⊓ ∃hasChild.Person)

MotherWithManyChildren ≡ ((Person ⊓ Female) ⊓ ∃hasChild.Person) ⊓ ≥3hasChild

GrandMother ≡ ((Person ⊓ Female) ⊓ ∃hasChild.Person) ⊓ ∃haschild.Parent

MotherWithoutDaughter ≡ ((Person ⊓ Female) ⊓ ∃hasChild.Person) ⊓ ∀hasChil.¬Woman

Wife ≡ (Person ⊓ Female) ⊓ ∃hasHusband.Man

FIG 6. L'extension de la TBox de la figure 5.

Par contre, les extensions uniques peuvent ne pas exister si une terminologie contient des cycles.

Néanmoins, il existe des terminologies avec cycles qui sont définitoriales. Par exemple, soit l'axiome

$$A \equiv \forall R.B \sqcup \exists R.(A \sqcap \neg A)$$

qui contient un cycle. Toutefois, comme $\exists R.(A \sqcap \neg A)$ est équivalent au concept bottom, l'axiome précédent est équivalent à l'axiome acyclique

$$A \equiv \forall R.B$$

D'autre part, on montre que toute terminologie ALC définitoriale est équivalente à une terminologie acyclique.

2.2.5. Les sémantiques du fixe point pour les terminologies cycliques :

Selon la sémantique que nous avons considérée plus haut, qui est essentiellement celle des logiques des prédicats du premier ordre, les terminologies sont définitoriales seulement si elles sont essentiellement acycliques. Les sémantiques du point fixe sont motivées par le fait qu'il existe des situations ou intuitivement des définitions cycliques sont significatives et l'intuition peut être capturée par les sémantiques du plus petit ou plus grand point fixe **[FRA 03]**.

Exemple : supposant qu'on veut définir le concept « homme qui n'a que des descendants male » : Momd. En particulier, un tel homme est Mos « homme qui n'a que des fils »

$$Mos \equiv Man \sqcap \forall hasChild.Man$$

Mais pour Momd, et d'après le filler du role hasChild, ici une definition recursive est nécessaire :

$$Momd \equiv Man \sqcap \forall hasChild.Momd$$

Afin d'atteindre le sens souhaité, nous devons interpréter cette définition en vertu d'une sémantique de point fixe appropriées. Nous allons montrer ci-dessous que la sémantique de plus grand point fixe capte notre intuition ici.

Les cycles apparaissent également lorsque l'on veut modéliser les structures récursives, par exemple, les arbres binaires :

$$BinaryTree \equiv Tree \sqcap \leq 2\ has\text{-}branch \sqcap \forall has\text{-}branch.BinaryTree$$

Comme la définition de Momd, une sémantique de point fixe donnera le sens souhaité. Toutefois, pour cet exemple, nous devons utiliser la sémantique du plus petit point fixe.

Nous allons maintenant donner une définition formelle de la sémantique du point fixe. Dans une terminologie T, chaque symbole de nom se produit exactement une fois dans le côté gauche d'un axiome $A \equiv C$. Cependant, nous pouvons considérer T comme une application de mapping qui associe à un symbole de nom A la description du concept $T(A) = C$. avec cette notation, une interprétation *I* est une modèle de T si et seulement si $A^I = (T(A))^I$. Cette caractérisation a la saveur de l'équation du point fixe. Nous exploitons cette similitude d'introduire une famille de mapping telles

qu'une l'interprétation est un modèle de T si et seulement si elle est un point fixe d'une tel mapping.

Soit T une terminologie, et soit *J* une base d'interprétation fixe de T. par Ext_J on note l'ensemble de toutes les extensions de *J*. Soit $T_J : \text{Ext}_J \rightarrow \text{Ext}_J$ l'application de mapping qui mappe l'extension *I* vers l'extension $T_J(I)$ définie par $A^{T_J(I)}=(T(A))^I$ pour chaque symbole de nom A.

Maintenant, *I* est le point fixe de T_J si et seulement si $I=T_J(I)$, i.e. $A^I=A^{T_J(I)}$ pour tout symbole nom de. Ca signifie que, pour chaque définition $A \equiv C$ dans T,

$A^I = A^{T_J(I)} = (T(A))^I = C^I$. Qui signifie que *I* est un modèle de T.

Proposition : soit T une terminologie, *I* une interprétation, et *J* la restriction de *I* pour les symboles de bases de T, alors *I* est un modèle de T SSI *I* est un point fixe de T_J.

Une terminologie T est définitoriale si et seulement si toute base interprétation *J* a une extension unique qui est le point fixe de T_J.

Pour avoir une idée de pourquoi les terminologies cycliques ne sont pas définitoriales, nous discuter de l'exemple de la terminologie qui se compose uniquement de l'Axiome Momd. Considérant la base d'interprétation *J* définie par :

$\Delta^J = \{\text{lehireche}^1, \text{lehireche}^2, \ldots.\} \cup \{\text{bourzig}^1, \ldots.., \text{bourzig}^{dernier}\}$,

$\text{Man}^J = \Delta^J$,

$\text{hasChild}^J = \{(\text{lehireche}^i, \text{lehireche}^{i+1}) \mid i \geq 1\} \cup \{(\text{bourzig}^i, \ldots.., \text{bourzig}^{i+1} \mid 1 \leq i < \text{dernier}\}$.

Cela signifie que la dynastie lehireche ne finie pas, alors il ya un dernier membre de la dynastie bourzig.

On veut identifier les points fixes de T_J^{Momd}. Noter qu'un individu sans fils est toujours dans l'interprétation de $(\forall \text{hasChild.Momd})^I$ peu importe Momd est interprété. Par conséquent, si *I* est une extension point fixe de *J*, alors $\text{bourzig}_{dernier}$ est dans $(\forall \text{hasChild.Momd})^I$ et aussi dans Momd^I. On conclue que chaque bourzig est un Momd. Soit I_1 l'extension de *J* tel que Momd^{I_1} comporte exactement la dynastie bourzig. Alors il est facile de vérifier que I_1 est un point fixe. Si, en plus de la dynastie bourzig, certains lehireche est un Momd, puis tous les membres de la dynastie lehireche avant et après lui doit appartenir au concept Momd. On peut facilement

vérifier que l'extension I_2 qui interprète Momd comme ensemble du domaine est également un point fixe, et qu'il n'ya pas d'autres point fixe.

Afin de donner un effet definitorial à une terminologie cyclique T, nous devons distinguer un point fixe du mapping T_J s'il ya plus d'un. Pour cela, nous définissons un ordre partiel « <= » sur les extensions de *J*. on dit que *I*<=*I*' si $A^I \subseteq A^{I'}$ pour chaque symbole de nom dans T **[FRA 03]**. dans l'exemple précédant, Momd est l'unique symbole de nom, alors $Momd^{I1} \subseteq Momd^{I2}$, nous avons $I_1 <= I_2$.

Un point fixe *I* de T_J est le plus petit point fixe si *I*<= *I'* pour chaque autre point fixe *I'*.

Un point fixe *I* de T_J est le plus grand point fixe si *I'*<= *I* pour chaque autre point fixe *I'*.

3 Descriptions du monde ABox :

Outre la terminologie ou la TBox, le second composant d'une base de connaissance est la description du monde ou la ABox.

3.1 Les assertions sur les individus :

La ABox décrit un état spécifique du domaine d'application en termes de concepts et de rôles. En fait, dans une ABox, on introduit des individus (en leur donnant des noms) ainsi que leurs propriétés. Soit C un concept et R un rôle. On peut former des assertions comme suit: C(a) et R(b; c).

La première assertion est dite assertion de concept, elle signifie que a appartient a (l'interprétation de) C tandis que la deuxième, qui est appelée assertion de rôle, elle signifie que c est un remplisseur du rôle R pour b.

MotherWithoutDaughter(MARY) Father(PETER)
hasChild(MARY, PETER) hasChild(PETER, HARRY)
hasChild(MARY, PAUL)

FIG 7. Une description du monde (Abox).

Prenant l'exemple de la figure 7, Father(PETER) signifie que PETER est un père et hasChild(MARY, PAUL) signifie que PAUL est enfant de MARY. Une ABOX notée A est un ensemble fini de telles assertions.

Une ABox peut être vu comme étant une instance d'une base de données relationnelle avec seulement des relations unaires et binaires. Toutefois, contrairement a la sémantique du monde clos des bases de données classiques, la sémantique des ABox est une sémantique monde ouvert étant donné que les systèmes de représentation de connaissances sont appliqués dans des situations ou l'on peut supposer que l'information est incomplète.

La sémantique des ABox est définie par l'extension des interprétations aux noms d'individus. Donc une interprétation ***I*=** $(\Delta^I, .^I)$ n'assigne pas uniquement des ensembles et des relations binaires aux concepts et au rôles mais aussi assigne a chaque nom d'individu *a* un élément a^i de Δ^I. On suppose que des noms d'individus distincts dénotent des objets différents. Par conséquent, l'interprétation doit satisfaire l'assomption de nom unique UNA (pour unique name assumption), i.e, si a et b sont des noms différents alors $a^i \neq b^i$.

D'autre part, une interprétation ***I*** satisfait l'assertion de concept C(*a*) si $a^i \in C^I$ Elle satisfait l'assertion de rôle R(a; b) si $(a^i, b^i) \in R^I$ et elle satisfait une ABox A si elle satisfait chaque assertion dans A. Dans ce cas, *I* est dite modèle de l'assertion ou de la ABox. Enfin, une interprétation *I* satisfait une assertion ou une ABox par rapport à une terminologie T si en plus d'être modèle de l'assertion ou de la base, elle est également modèle de la terminologie T.

3.2 Les noms d'individus dans les langages de description :

Parfois, il est convenable pour permettre les noms individuels (également appelé nominales) non seulement dans le ABox, mais aussi dans le langage de description. Certains constructeurs de concept employant les individus produire dans les systèmes, parmi ces constructeurs on trouve le constructeurs « ensemble » ou « on-of » écrit :

$$\{a_1, \ldots, a_n\},$$

Ou $a_1, \ldots, a_n$ sont des noms d'individus, un tel concept est interprété par :

$\{a_1, \ldots, a_n\}^I = \{a_1{}^I, \ldots, a_n{}^I\}$,

Avec les ensemble dans le langage de description on peut maintenant défini le concept des membres permanents de conseil de sécurité de UN {la chine, la France, la Russie, UK, US}.

Un autre constructeur utilise les noms d'individus est le constructeur « remplisseur » ou « filler » : R : a, pour un rôle R la sémantique de ce constructeur est défini par :

$$(R : a)^I = \{d \in \Delta^I \mid (d, a^I) \in R^I\},$$

On dit la que a est le filler du rôle R. Dans les langages de description avec le constructeur « ensemble de » et la quantification existentielle, le constructeur « filler » n'ajoute rien parce que : R : a et $\exists$R. {a} sont équivalents.

4. Conclusion :

Ce chapitre a pour objet une introduction à la logique de description. Après une description du langage de base *AL* et ses dérivés, nous avons présenté ce que sont les TBox et les ABox. Ainsi, et comme n'importe quel autre ensemble d'axiomes, il contient des connaissances implicites qui peuvent être explicités par des inférences. Ce qui nous allons aborder dans le chapitre suivant, le raisonnement.

Chapitre III

Le raisonnement dans les DL

1. introduction :

Un système de représentation des connaissances basées sur les logiques de Descriptions est capable d'effectuer certains types de raisonnement. Comme dit précédemment, l'objectif d'un système de représentation des connaissances va au-delà de stockage des définitions de concepts et d'assertions. Une base de connaissances dans les DLs contient une TBox et une ABox à une sémantique qui en fait l'équivalent d'un ensemble d'axiomes dans la logique des prédicats du premier ordre. Ainsi, comme n'importe quel autre ensemble d'axiomes, il contient des connaissances implicites qui peuvent être explicités par des inférences. Pour par exemple, de la TBox à la figure 5 et le ABox dans la figure 7, on peut conclure que Marie est une grand-mère, même si cette connaissance n'est pas explicitement mentionné comme une assertion.

Dans ce qui suit, nous allons discuter de ces inférences, premièrement pour les concepts, puis pour TBoxes et ABoxes, et enfin pour TBoxes et ABoxes ensemble. Il s'avérera qu'il ya un problème d'inférence principal, à savoir la vérification de consistance pour ABoxes, à laquelle toutes les autres problèmes d'inférences peuvent être réduites. Ensuite nous allons présenter deux types des algorithmes de raisonnement dans les DL à savoir les algorithmes structurels et les algorithmes logiques.

2. Tâches de raisonnement pour la TBox :

Lors de la modélisation d'un domaine, un ingénieur de la connaissance construit une terminologie, disons T, en définissant de nouveaux concepts, éventuellement en fonction des autres qui ont été définis auparavant.

Au cours de ce processus, il est important de savoir si un concept nouvellement défini a un sens ou si elle est contradictoire. D'un point de vue logique, un concept a un sens

pour nous s'il ya une certaine interprétation qui satisfait les axiomes de T (qui est un modèle de T) tel que le concept désigne un ensemble non vide dans cette interprétation. Un concept ayant cette propriété est satisfiable par rapport à T et non satisfiable autrement.

Vérifier la satisfiabilité d'un concept est une inférence clé, un nombre important d'inférences peut être réduit au test de satisfiabilité.

Afin de vérifier si un modèle de domaine est correct, ou pour optimiser les requêtes qui sont formulées comme des concepts, on peut vouloir savoir si un concept est plus général qu'un autre: c'est le problème de subsomption. Un concept C est subsumé par un concept D si, dans tous les modèles de la terminologie T l'ensemble désigné par C est un sous-ensemble de l'ensemble désigné par D. Les algorithmes de subsomption sont également utilisés pour organiser les concepts d'une TBox dans une taxonomie en fonction de leur généralité. D'autres relations intéressantes entre les concepts sont l'équivalence et la disjonction.

Ces propriétés sont officiellement définies comme suit. Soit T une TBox :

Satisfiabilité: Un concept C est satisfaisable par rapport à T s'il existe un modèle I de T tel que C^I n'est pas vide. Dans ce cas nous disons aussi que *I* est un modèle de C.

Subsomption : un concept C est subsumé par un concept D par rapport à T si $C^I \subseteq D^I$ pour tout modèle *I* de T. dans ce cas on écrit $C \sqsubseteq_T D$ ou $T \models C \sqsubseteq D$.

Equivalence : deux concepts C et D sont équivalent par rapport a T si $C^I = D^I$ pour tout modèle *I* de T. dans ce cas on écrit $C \equiv_T D$ ou $T \models C \equiv D$.

Disjonction : deux concepts C et D sont disjoints par rapport a T si $C^I \cap D^I = \emptyset$ pour tout modèle *I* de T.

Dans le cas particulier où la TBox est vide on écrit simplement $\models C \sqsubseteq D$ si C est subsumé par D et $\models C \equiv D$ si C et D sont équivalents.

Traditionnellement, le mécanisme de raisonnement de base fournis par les systèmes DL vérifié la subsomption de concepts. C'est, en fait, est suffisante pour mettre en œuvre aussi les autres inférences, comme on le voit par les réductions suivantes :

Réduction vers la subsomption : pour deux concepts C et D on a :

(i) *C est non satisfiable* $\Leftrightarrow$ *C est subsumé par* $\perp$ *;*
(ii) *C et D sont équivalents* $\Leftrightarrow$ *C est subsume par D et D est subsumé par C;*
(iii) *C et D sont disjoints* $\Leftrightarrow$ $C \sqcap D$ *est subsumé par* $\perp$.

Si, en plus d'intersection, le système permet de former la négation d'une description, on peut réduire la subsomption, l'équivalence, et la disjonction des concepts en un problème de satisfiabilité :

Réduction vers la satisfiabilité :

(i) *C est subsumé par D* $\Leftrightarrow$ $C \sqcap \neg D$ *est non satisfiable;*
(ii) *C et D sont équivalents* $\Leftrightarrow$ $(C \sqcap \neg D)$ *and* $(\neg C \sqcap D)$ *sont non satisfiables;*
(iii) *C et D sont disjoints* $\Leftrightarrow$ $C \sqcap D$ *est non satisfiable.*

Afin d'obtenir des procédures de décision pour toute les quatre inférences dont nous avons discuté, il suffit de développer des algorithmes qui décider la satisfiabilité des concepts, à condition que le langage soutient la négation des concepts arbitraires. En fait, cette observation a motivé les chercheurs à étudier les langages de description dans lequel, pour chaque concept, on peut aussi former la négation de ce concept **[SMO 88; SCH 91; DON 91;97].** L'approche qui tient compte de la satisfiabilité est l'inférence principale a donné lieu à un nouveau type d'algorithmes pour le raisonnement en logiques de description, qui peut être comprise comme tableaux de calcule spécialisés.

La proposition suivante montre que la non satisfiabilité est un cas particulier de chacun des autres problèmes :

Soit C un concept, alors ces phrases sont équivalentes :

(i) *C est non satisfiable;*
(ii) *C est subsumé par* $\bot$ *;*
(iii) *C et* $\bot$ *sont équivalents;*
(iv) *C and* $\top$ sont *disjoins.*

3. L'élimination du TBox :

Dans les applications, les concepts viennent habituellement dans un contexte d'une TBox. Toutefois, pour l'élaboration de procédures de raisonnement, il est conceptuellement plus facile de faire abstraction de la TBox ou, ce qui revient au même, à supposer qu'elle est vide.

Si T est une TBox acyclique, on peut toujours réduire les problèmes de raisonnement par rapport à T à des problèmes par rapport a une TBox vide. Rappelons que dans l'expansion chaque définition est de la forme A≡D tel que D ne contient que des symboles de base, pas des symboles de noms. Maintenant, pour chaque concept C nous définissons l'expansion de C par rapport à T comme le concept C' qui est obtenu à partir de C en remplaçant chaque occurrence de A dans C par le concept D où: A≡D

est la définition de A dans T', l'expansion de la T.

Par exemple, nous obtenons l'extension du concept Woman ⊓ Man (dans la TBox de figure 5) en considérant son extension (figure 6), et remplacer les concepts Woman et Man par les cotés droites de leurs définitions dans cette extension. Le concept résultant c'est : Person ⊓ Female ⊓ Person ⊓ ¬(Person ⊓ Female).

Maintenant, C est satisfiable par rapport a T si C' est satisfiable par rapport a T, cependant C' ne contient pas des symboles définis, Alors **C' est satisfiable par rapport a T s'il est satisfiable.**

Si D est un autre concept, et on a aussi D≡D'. Alors $C \equiv_T D$ SSI $C' \equiv_T D'$, et $C \sqsubseteq_T D$ SSI $C' \sqsubseteq_T D'$. C' et D' contiennent que des symboles de base, cela implique :

- $T \models C \sqsubseteq D$ SSI $\models C' \sqsubseteq D'$
- $T \models C \equiv D$ SSI $\models C' \equiv D'$

4. Tâches de raisonnement pour la Abox :

Après que l'ingénieur cogniticien a conçu une terminologie et a utilisé les services de raisonnement du système DL pour vérifier que tous les concepts sont satisfiable et que les relations de subsomption sont maintenues, l'ABox peut être rempli avec des assertions sur les individus. Nous rappelons qu'une ABox contient deux sortes d'assertions, assertions des concepts de la forme C(a) et assertions des rôles de la forme R(a, b). Bien sûr, la représentation de ces connaissances doit être consistance, parce que sinon -du point de vue de la logique - on pourrait tirer des conclusions arbitraires de lui. Si, par exemple, la ABox contient les assertions Mother(Marie) et Father(Marie), le système devrait être en mesure de savoir que, à la fois avec la TBox de famille, ces déclarations sont non contradictoires.

Une ABox A est consistante par rapport a une TBox T, s'il ya une interprétation qui est un modèle de A et T. Nous disons simplement que A est consistante si elle est consistante par rapport a une TBox vide.

Par exemple, l'ensemble des assertions {Mother (Marie), Father (Marie)} est consistant (par rapport a une TBox vide), parce que sans autres restrictions sur l'interprétation du Father et Mother, les deux concepts peuvent être interprétées de telle façon dont ils ont un élément commun. Toutefois, ces assertions ne sont pas consistantes par rapport à la TBox de famille, puisque dans chaque modèle de celui-ci, Mother et Father sont interprétée comme disjoints.

De même que pour les concepts, vérifier la consistance d'une ABox par rapport a une

TBox acyclique peut être réduite à une vérification d'une ABox étendue. Nous définissons l'extension de A par rapport à T comme une ABox A' qui est obtenu à partir de A en remplaçant chaque assertion C(a) en A par l'assertion C'(a), où C' est l'extension de C par rapport à T. Dans tous les modèles de T, un concept C et son extension C' sont interprétés de la même manière. Par conséquent, A' est consistante par rapport a T si et seulement si A est consistante. Toutefois, puisque A' ne contient pas des symboles définis dans T, elle est consistance par rapport a T si et seulement si elle est consistance. Nous concluons :

- A est consistance par rapport a T si A' est consistance.

Dans une ABox A, on peut se poser des requêtes sur les relations entre les concepts, les rôles et les individus. L'inférence prototypique sur lequel ces requêtes sont basées est l'instanciation. Nous disons que l'assertion α est entraînée par A et nous écrivons A |= α, si toutes les interprétations qui satisfait A, qui est un modèle de A, satisfait également α. Si α est une assertion de rôle, l'instanciation est simple, dés lors que notre langage de description ne contient pas des constructeurs pour former des rôles complexes.

Si α est de la forme C(a), on peut réduire l'instanciation a un problème de consistance de ABox parce que :

- A |= C(a) SSI A ∪ {¬C(a)} est non consistante.

La satisfiabilité d'un concept peut être réduite à un test de consistance parce que pour chaque concept C nous avons :

- C est satisfiable SSI {C(a)} est consistante.

Pour les applications, des inférences plus complexes que la consistance et l'instanciation sont habituellement exigées. Le problème de récupération (recouvrement) est, étant donné une ABox A et un concept C, alors trouver tous les individus a telle que A |= C (a). Un algorithme non optimisé peut être réalisée par des tests pour chaque individu se produisant dans la ABox s'il est une instance du C.

L'inférence inverse de la récupération est le problème de la réalisation: étant donné un individu a et un ensemble de concepts, trouver le concept le plus spécifique de l'ensemble tel que A |= C (a). Ici, les concepts les plus spécifiques sont ceux qui sont minimes par rapport à l'ordre de subsomption. Réalisation peut, par exemple, être utilisé dans les systèmes qui génèrent le langage naturel si les termes sont indexés par

des concepts et si un terme précise que possible doit être trouvé pour un objet se produisant dans un discours.

5. Les algorithmes de raisonnement :

On a vu que les problèmes d'inférence pertinents peuvent être réduits au problème de consistance de la Abox, en sachant que le langage de description en question permet la négation et la disjonction. Cependant, les premiers langages de description et aussi certains des systèmes DL actuels ne le permettent pas, Pour ces logiques de description, la subsomption des concepts peut être utilisés par qui ont dits algorithmes de subsomption structurelle, à savoir, les algorithmes qui permettent de comparer la structure syntaxique des (aussi normalisé) descriptions des concepts. Mais ces derniers ne sont complets que pour les langues assez simples avec une moindre expressivité. En particulier, les logiques de descriptions avec la négation et la disjonction ne peuvent pas être traitées par des algorithmes de subsomption structurelle, pour ces langages, les algorithmes qui ont dits algorithmes basés tableaux se sont avérés être très utile. Dans le domaine de logiques de descriptions, le premier algorithme basé tableaux a été présenté par **[SCH 91]** pour la satisfiabilité des concepts ALC. Depuis, cette approche a été utilisée pour obtenir des algorithmes de satisfiabilité saines et complète (et donc aussi des algorithmes de subsomption) pour une grande variété des logiques de description qui étendent ALC.

Intuitivement, au lieu de concevoir de nouveaux algorithmes en DLs, on peut essayer de réduire le problème qu'on veut résoudre à un problème d'inférence connu en logique classique. Par exemple, la décidabilité du problème d'inférence en logique *ALC* peut être obtenue en ayant la décidabilité du langage *L*2 (un fragment de la logique des prédicats du premier ordre à deux variables) en sachant que tout concept *ALC* peut être traduit dans le langage *L*2. Prenons l'exemple suivant :

Une traduction directe du concept $\forall R.(\exists R.A)$ génère la formule :

$$\forall y.(R(x, y) \rightarrow (\exists z.(R(y, z) \wedge A(z)))).$$

Comme la sous formule $\exists z.(R(y, z) \wedge A(z))$ ne contient pas la variable x, cette dernière peut être réutilisée à la place de z. Ce renommage génère la formule équivalente suivante :

$\forall y.(R(x, y) \rightarrow (\exists x.(R(y, x) \wedge A(x))))$ qui utilise uniquement deux variables.

Toutefois, la complexité des procédures de décision obtenues ainsi est souvent plus

élevée qu'il en faut : par exemple, le problème de satisfiabilité de la logique *L*2 est NExpTime-complet alors que le problème de satisfiabilité des descriptions des concepts *ALC* est "uniquement" PSpace-complet. Ceci montre l'intérêt de développer de nouveaux algorithmes spécifiques aux logiques de description.

5.1 Algorithmes de subsomption structurelle (NC):

Ces algorithmes opèrent en général en deux phases :

1. la Normalisation des descriptions en question ;
2. la Comparaison des structures syntaxiques des formes normales.

En vue de simplifier les choses, on présente cette approche dans le cadre du langage simple FL_0. Ensuite, on montre comment traiter le concept bottom ($\perp$), la négation atomique ($\neg A$) et les restrictions de nombres ($\leq nR$ et $\geq nR$).

La description d'un concept FL_0 est sous forme normale si et seulement si elle est de la forme : $A_1 \sqcap \ldots \sqcap Am \sqcap R_1.C_1 \sqcap \ldots \sqcap Rn.Cn$

Ou $A_1 \ \ldots \ Am$ sont des concepts de noms distincts, $R_1 \ \ldots \ Rn$ sont des noms de rôles distincts, et $C_1 \ldots Cn$ sont des descriptions de concepts FL_0 sous forme normale.

Il est à noter que toute description peut être transformée en une forme normale en utilisant l'associativité, la commutativité, l'idempotence de $\sqcap$.

Proposition : Soient

$$A_1 \sqcap \ldots \sqcap Am \sqcap R_1.C_1 \sqcap \ldots \sqcap Rn.Cn$$

La forme normale de la description du concept FL_0 C, et

$$B_1 \sqcap \ldots \sqcap Bm \sqcap S_1.D_1 \sqcap \ldots \sqcap Sn.Dn$$

La forme normale de la description du concept FL_0 D.

Alors, C $\sqsubseteq$Dsi et seulement si les deux conditions suivantes sont vérifiées :

1. pour tout i, $1 \leq i \leq k$, il existe j, $1 \leq j \leq m$ tel que Aj$\sqsubseteq$ Bi
2. pour tout i, $1 \leq i \leq l$, il existe j, $1 \leq j \leq n$ tel que Si = Ri et Cj$\sqsubseteq$ Di

Cette caractérisation de subsomption est saine et complète. Elle permet de définir un algorithme récursif polynomial **[LEV 87].**

Soient quatre concepts A, B, C et D ou A$\sqsubseteq$ B et on veut montrer par l'algorithme de subsomption que : A $\sqcap \forall$R.C $\sqcap\forall$R.D$\sqsubseteq$ B $\sqcap\forall$R.C

Normalisation : A $\sqcap \forall$R.C $\sqcap\forall$R.D$\equiv$ A $\sqcap \forall$R.(C $\sqcap$D)

Donc ¸ca revient à montrer que : A ⊓ ∀R.(C ⊓D) ⊑ B ⊓∀R.C

Comparaison structurelle : i.e, on vérifie que pour chaque sous expression X du membre droit B ⊓∀R.C il existe une sous expression Y du membre gauche A ⊓ ∀R.(C ⊓D) tel que Y ⊑X :

– X = B, Y = A avec A⊑ B

– X = ∀R.C, Y = ∀R.(C⊓ D) : il faut montrer que C⊓ D⊑C.

– Donc on applique le même algorithme : X = C, Y = C⊓ D avec C⊑ C.

Il est possible d'étendre cet algorithme afin de tester la subsomption de descriptions de concepts appartenant au langage FL_0 enrichi par le concept bottom et la négation atomique. Nous détaillons un algorithme NC pour le langage de description de concepts et de rôles ALNR, Voici les différentes règles de comparaison associées au test D ⊑C dans l'algorithme de subsomption **[NAP 97].**

- [bottom] : si D = BOTTOM alors retourner vrai.
- [concept primitif]: si C et si D dénotent des concepts primitifs, alors tester

  C = D ou D⊑ C (la relation D ⊑C peut être vérifiée par transitivité).
- [rôle primitif] : si C et si D dénotent des rôles primitifs, alors tester

  C= D ou D⊑C (la relation D⊑ C peut être vérifiée par transitivité).
- [¬] : si C = (¬C') et D = (¬D'), où C' et D' dénotent des concepts primitifs, alors tester C' = D' ou C'⊑D'.

  Les cas où les négations sont croisées, qui consistent à comparer, C =(¬C') et D, ou C et D = (¬D') ne sont pas traités.
- [∀]: si C = (∀ R.C) et si D = (∀ S.D), alors tester R⊑S et D⊑C.
- [∃]: si C = (∃R) et si D = (∃S), alors tester S⊑R.
- [≥] : si C = (≥nR) et si D = (≥mS), alors tester n ≤m et S⊑R.
- [≤] : si C = (≤nR) et si D = (≤mS), alors tester m≤n et R⊑S.
- [D est une conjonction] : si D = (D_1 ⊓… ⊓ D_n) dans [concept primitif], [∀], [∃], [≥] et [≤], alors tester $D_i ⊑ C_j$ pour un 1≤i≤ n, et 1≤j≤m.

Enfin, on trouve dans **[BOR 94]** un algorithme de subsomption structurelle pour le

langage *ALN*. Au delà de ce dernier, les algorithmes de subsomption structurelle se trouvent incomplets. En particulier, ils ne traitent pas la disjonction, la négation complète et le quantificateur existentiel complet d'où l'utilité des algorithmes de tableaux qui dont l'objet de la section suivante.

5.2 Algorithmes de tableaux :

Les algorithmes de tableaux réduisent le problème de subsomption au problème de satisfiabilité. En fait, on sait que $C \sqsubseteq D$ si et seulement si $C \sqcap \neg D$ est non satisfiable.

Avant de présenter un algorithme de tableaux pour la logique *ALCN*, on illustre le principe correspondant via l'exemple suivant :

Soient A et B deux noms de concepts, R un nom de rôle et supposons que l'on veut savoir si :

$$(\exists R.A) \sqcap (\exists R.B) \sqsubseteq \exists R.(A \sqcap B).$$

Ceci revient à tester la non satisfiabilité de

$$(\exists R.A) \sqcap (\exists R.B) \sqcap \neg (\exists R.(A \sqcap B)).$$

Dans un premier temps, on déplace la négation en utilisant les règles de Morgan et les règles usuelles de quantificateurs. Comme résultat on obtient la description

$$C = (\exists R.A) \sqcap (\exists R.B) \sqcap \forall R.(\neg A \sqcup \neg B)).$$

Qui est sous forme NNF (pour negation normal form), i.e., la négation se trouve uniquement devant les noms de concepts.

Ensuite, on essaye de construire une interprétation finie ***I*** telle que $C^I \neq \emptyset$.

Ceci signifie qu'il doit exister un individu de Δ^i *qui soit un élément de* C^I.

Soit b un tel élément et donc b doit satisfaire les contraintes $b \in (\exists R.A)^I$,

$b \in (\exists R.B)^I$ et $b \in (\forall R.(\neg A \sqcup \neg B)^I$.

De $b \in (\exists R.A)^I$ *on déduit qu'il doit exister un élément c tel que* $(b, c) \in R^I$ et $c \in A^I$. *Il est de même,* $b \in (\exists R.B)^I$ *on déduit qu'il doit exister un élément d tel que* $(b, d) \in R^I$ et $d \in B^I$.

- *Pour toute restriction existentielle, on introduit un nouvel individu comme remplisseur du rôle. En plus, cet individu soit satisfaire les contraintes correspondantes.*

Comme b doit également satisfaire $b \in (\forall R.(\neg A \ \mathbf{U} \ \neg B)^I$ et c et d sont des R-remplisseurs de b, on tire que $c \in (\neg A \ \mathbf{U} \ \neg B)^I$ et $d \in (\neg A \ \mathbf{U} \ \neg B)^I$.

- *On utilise les restrictions de valeurs afin d'imposer de nouvelles contraintes sur les individus.*

D'autre part, $c \in (\neg A \ \mathbf{U} \ \neg B)^I$ signifie que $c \in (\neg A)^I$ ou $c \in (\neg B)^I$. Toutefois, $c \in (\neg A)^I$ contredit l'autre contrainte $c \in (A)^I$ si bien qu'on doit opter pour $c \in (\neg B)^I$ I. D'une manière similaire, on prend $d \in (\neg A)^I$.

- *Pour les contraintes disjonctives, on choisit les possibilités successivement en faisant des retours arrière à chaque fois qu'une contradiction est découverte.*

Donc là on vient de satisfaire toutes les contraintes en générant une interprétation I telle que :

$\Delta^i = \{b, c, d\}$, $R^I = \{(b, c), (b, d)\}$, $A^I = \{c\}$ *et* $B^I = \{d\}$, *i.e., C est satisfiable et donc* $(\exists R.A) \sqcap (\exists R.B)$ *n'est pas subsumé par* $\exists R.(A \sqcap B)$.

Maintenant, on rajoute une restriction de nombres au premier concept, i.e., on veut vérifier si

$$(\exists R.A) \sqcap (\exists R.B) \sqcap \leq 1R \sqsubseteq \exists R.(A \sqcap B).$$

Qui l'est intuitivement.

En premier lieu, on procède exactement comme précédemment mais la pour satisfaire la nouvelle contrainte $b \in \leq 1R$, *les individus c et d doivent être identiques.*

- *Donc si une restriction de nombres est violée on doit identifier les différents individus remplisseurs de rôle.*

Dans ce cas, $c = d \in A^I$, $c = d \in B^I$ et $c = d \in (\neg A \ \mathbf{U} \ \neg B)^I$ produit toujours un conflit si bien que le concept C soit non satisfiable et on conclut que $(\exists R.A) \sqcap (\exists R.B) \sqcap \leq 1R \sqsubseteq \exists R.(A \sqcap B)$.

Afin de représenter les contraintes de type "*a* appartient à l'interprétation de *C*" et "*b* est un R-remplisseur de a", on peut prendre les assertions ABox comme structures de données. Mais dans ce cas on élimine l'hypothèse de nom unique (UNA) pour de

telles ABox car comme on l'a vu dans l'exemple précédent, on peut être amenés à identifier des noms d'individus différents lorsqu'il s'agit de restriction de nombres.
En revanche, on permet des assertions d'inégalité explicites de la forme $a \neq b$ qui est satisfaite par une interprétation *I* si et seulement si $a^I \neq b^I$.
Plus formellement, soit *C* un concept ALCN sous forme NNF. Afin de tester la satisfiabilité de ce dernier, l'algorithme de tableaux démarre avec une ABox $A_0 = \{C(x)\}$ et applique les règles de transformations qui préservent la consistance jusqu'a ce qu'aucune règle ne puisse être appliquée.
Si la ABox "complète" obtenue à la fin ne contient pas de contradiction (appelée clash), alors A_0 est consistante (et donc *C* est satisfiable) sinon elle est non consistante (non satisfiable). La ABox A contient un conflit si l'une des trois situations suivantes se produit :

i. $\{\perp(x)\} \subseteq$ A pour un nom d'individu x,
ii. $\{C(x), \neg C(x)\} \subseteq$ A pour un nom d'individu x et un nom de concept C,
iii. $\{(\leq n\ R)(x)\}$et le nombre de R-successeur de x est strictement plus grand que n
iv. $\{(\leq n\ R)(x), (\geq m\ R)(x)\}$ ou n<m.

Par ailleurs, les règles de transformations liées à la disjonction et à la restriction de nombres ne sont pas déterministes. Pour cela, on considère des ensembles finis de ABox $S = \{A_1, \ldots, A_k\}$ où un ensemble est consistent si et seulement s'il existe *i*, $1 \leq i \leq k$, tel que *Ai* soit consistent.
Donc, une règle de transformation est appliquée a un ensemble fini de Abox *S* comme suit : elle prend un élément *A* de *S* et le remplace par une seule Abox *A'*, par deux ABox *A'* et *A''*, ou bien par un nombre fini de ABox $A_{i,j}$.

Les règles de transformations sont résumées comme suit :

La règle- $\sqcap \rightarrow$

Condition : A contient $(C_1 \sqcap C_2)(x)$, $C_1(x)$ et $C_2(x)$ ne sont pas toutes deux dans A.

Action : A'=A $\cup$ $\{C_1(x), C_2(x)\}$.

La règle- $\sqcup \rightarrow$

Condition : A contient $(C_1 \sqcup C_2)(x)$, ni $C_1(x)$ ni $C_2(x)$ ne sont dans A.

Action : A'=A $\cup$ $\{C_1(x)\}$, ou A''= A $\cup$ $\{C_2(x)\}$.

La règle-$\exists \rightarrow$

Condition : A contient (∃R.C)(x), mais il n'y a pas de nom d'individu z tel que C (z) et R (x, z) sont dans A.

Action : A'=A ∪ {C(y), R(x, y)} où y est un nom d'individu ne se produisent pas dans A.

La règle-∀ →

Condition : A contient (∀ R.C)(x), et R(x, y), mais ne contient pas C(y),

Action: A'=A ∪ {C(y)}.

La règle-≥ →

Condition : A contient (≥nR)(x), et il n'existe pas des individus z_1, …. , z_n

Tel que R(x, z_i) ($1 \leq i \leq n$) et $z_i \neq z_j$ ($1 \leq i<j \leq n$) sont dans A.

Action : A'=A ∪ {R(x, y_i) | $1 \leq i \leq n$} ∪ { $y_i \neq y_j$ | $1 \leq i<j \leq n$ },

Ou y_1, …. , y_n sont des individus distincts n'occurrent pas dans A.

La règle-≤ →

Condition : A contient des noms d'individus distincts y_i, …, y_{n+1} tel que (≤nR)(x) et R(x, y_i),…, R(x, y_{n+1}) sont dans A, et $y_i \neq y_j$ n'est pas dans A pour tout $i \neq j$.

Action : pour tout pair y_i, y_j tel que i>j et $y_i \neq y_j$ n'est pas dans A, la Abox

$A_{i,j}$ = [y_i / y_j] A est obtenue à partir de A par le remplacement de chaque occurrence

De y_i par y_j .

6. Conclusion :

Puisque les logiques de description sont un formalisme de représentation de connaissances (RC) et que les systèmes de RC doivent répandre aux questions d'utilisateur dans un temps raisonnable. A la différence de la logique du premier ordre, les procédures de décision doivent toujours terminer, pour les réponses positives et même négatives. Puisque la garantie de réponse dans un temps fini n'implique pas que la réponse est donnée dans un temps raisonnable, l'étude de complexité des DL avec des problèmes d'inférences décidables est un enjeu important.

Décidabilité et complexité de problèmes d'inférence dépend de pouvoir expressif des logiques de description, d'une part, les DL les plus expressives ont des problèmes d'inférences de haute complexité, ou ils peuvent même être non décidable.

D'autre part, les logiques de description légère (avec des procédures de raisonnement efficaces) ne sont pas suffisamment expressives pour représenter suffisamment les concepts importants de l'application donnée. L'étude de l'enjeu entre l'expressivité des DL et la complexité de ses procédures de raisonnement a été un des problèmes importants dans la recherche des DL.

Chapitre IV

Etude et réalisation d'un raisonneur dans les logiques de descriptions

1. Introduction :

Afin d'atteindre notre objectif de réaliser un raisonneur DL, nous avons développé notre application avec le langage java, qui est un langage de programmation orienté objet, pour cela nous avons utilisé l'environnement de développement intégré Borland Jbuilder 8.

Premièrement, nous avons conçu notre propre syntaxe qui permet de créer les entrées (des fichiers texte) de l'application : les TBox et les ABox, cette syntaxe est résumée comme suite :

La conjonction (⊓) : est codifiée avec le caractère &.

La disjonction (**U**) : est codifiée avec le caractère │.

La quantification universelle (∀): est codifiée avec le caractère @.

La quantification existentielle (∃): est codifiée avec le caractère #.

La négation (¬): est codifiée avec le caractère ^.

Pour décrire que l'individu b est une instance du concept C, nous avons met

b : C.

Pour décrire que les individus b et c sont reliés par un rôle R, nous avons met

b R c.

Nous avons travaillé avec un langage minimum qui est le langage *ALCN (voir chapitre 2).*

Voici un exemple d'une ABox :

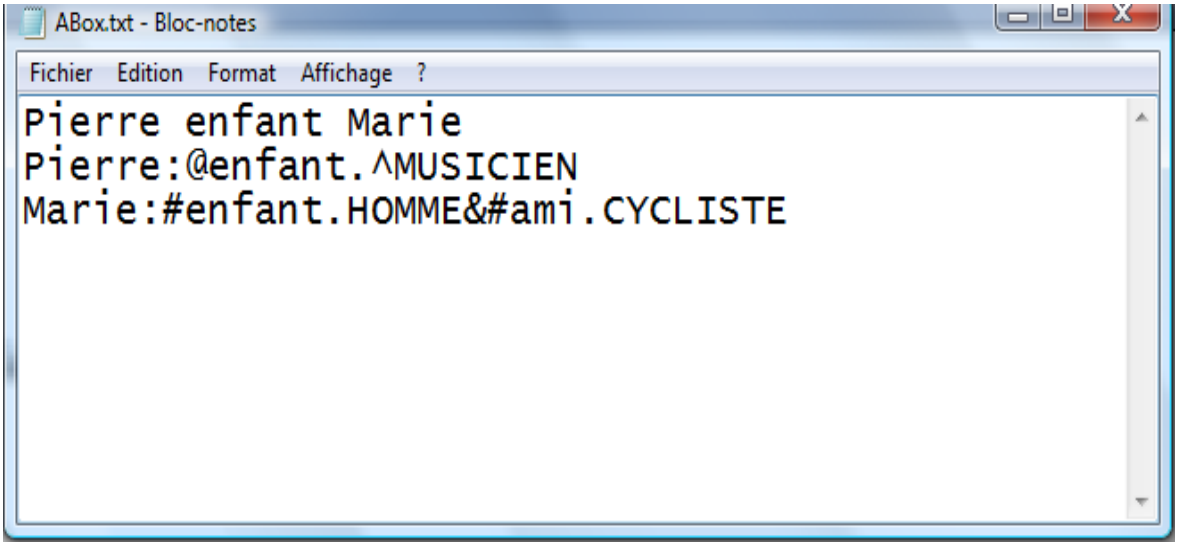

FIG 8. Exemple d'une ABox avec la syntaxe utilisée.

2. Présentation de l'application développée :

Voici l'interface de notre raisonneur :

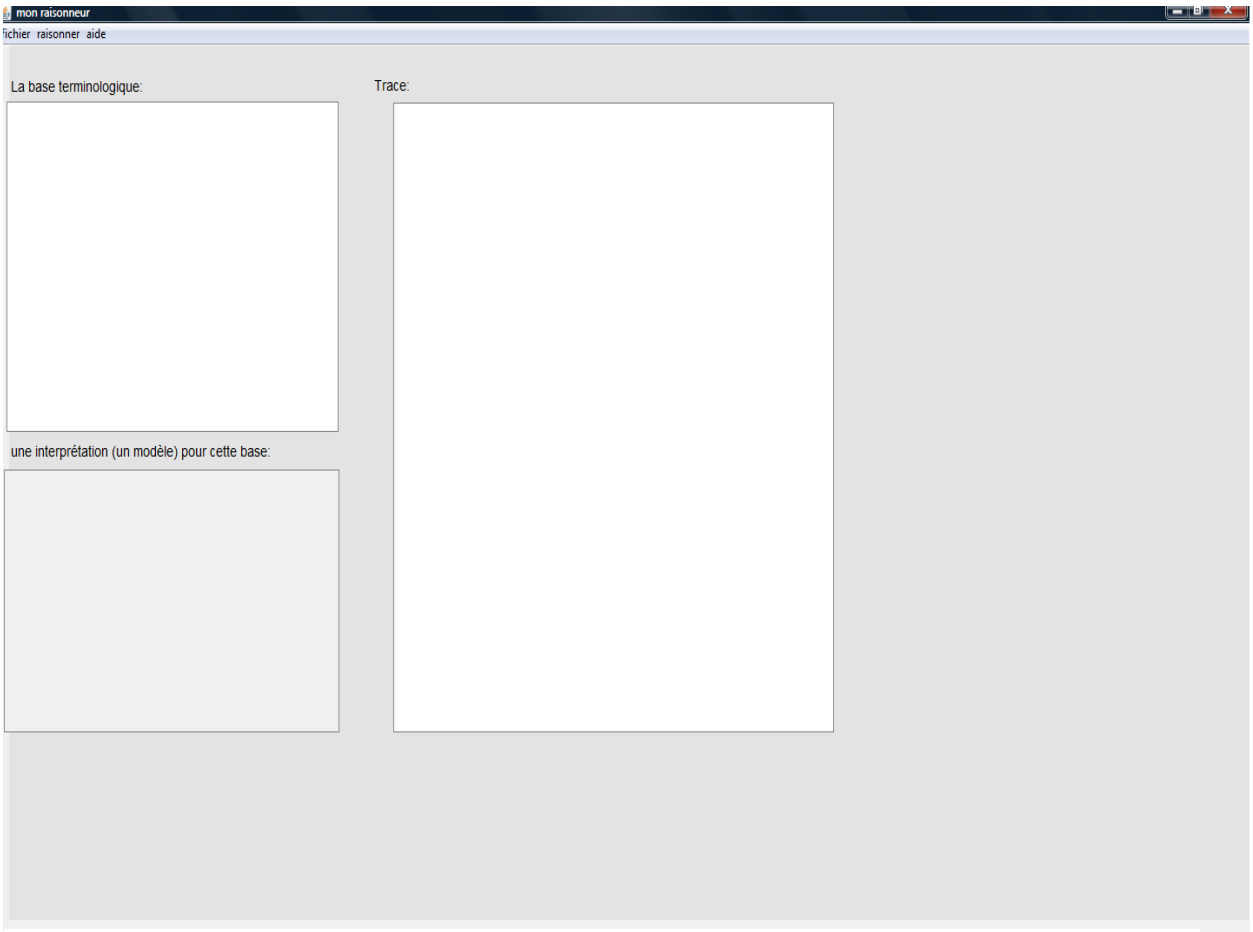

FIG 9. L'interface du raisonneur.

Qui contient une barre de menus, une zone pour afficher l'entrée en cour, une zone pour afficher l'interprétation d'une base si elle est consistante, et une zone pour

afficher la trace des algorithmes développés.

La barre de menus contient :

Le menu Fichier :

- Le sous menu : ouvrir une TBox : pour ouvrir une TBox (terminologie).
- Le sous menu : ouvrir une ABox : pour ouvrir une ABox.

Les TBoxs et les ABoxs représentent ici les entrées de l'application.

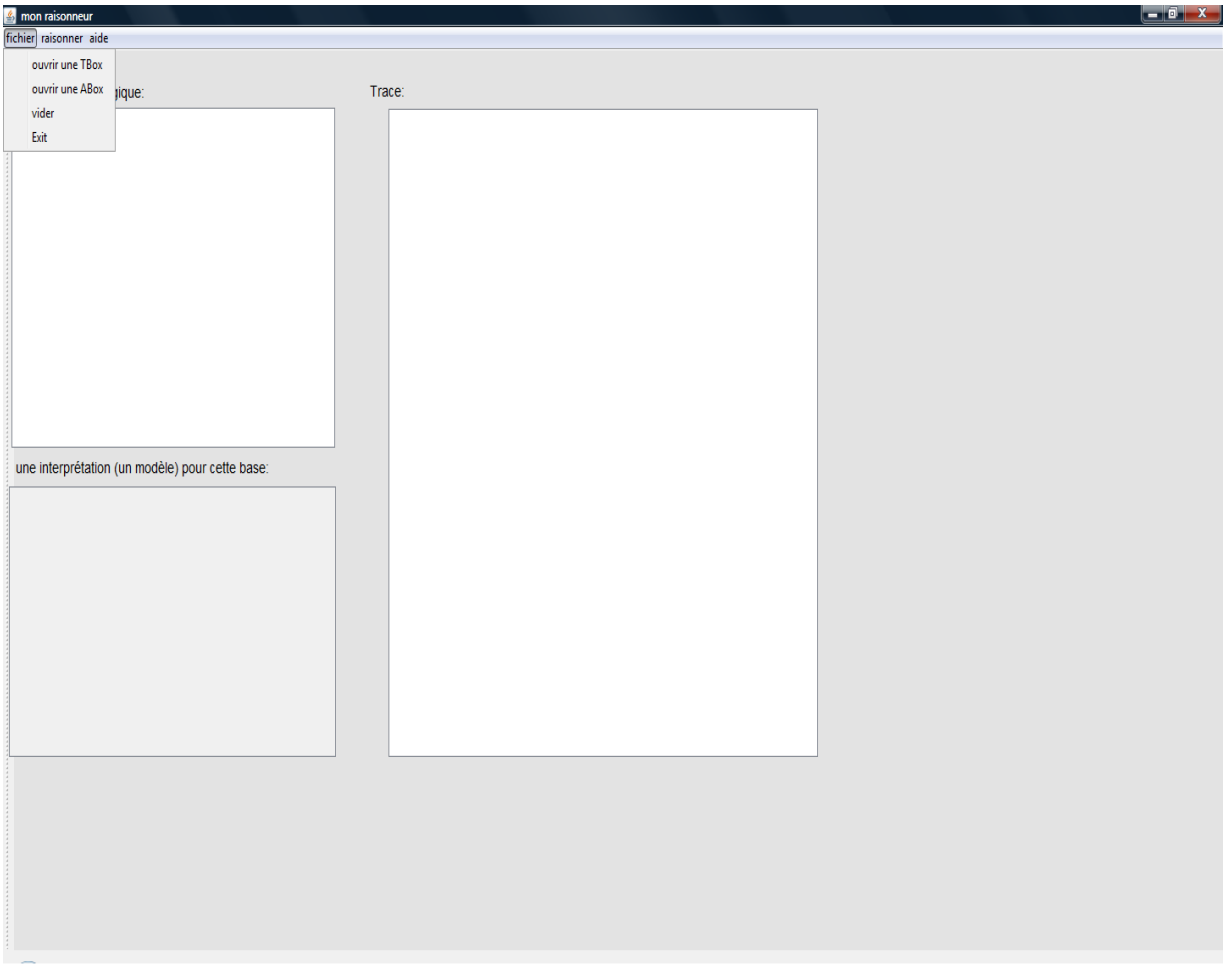

FIG 10. Le sous menu fichier.

Le menu raisonner :

- Le sous menu teste de consistance : pour tester la consistance d'une base.
- Le sous menu teste d'instanciation : pour tester si un individu donné b appartient a l'interprétation d'un concept donné C.
- Le sous menu teste de subsomption : pour tester si un concept donné C est subsumé par un autre concept donné D.
- Le sous menu interroger la base : pour poser des questions concernant le contenu de la base.

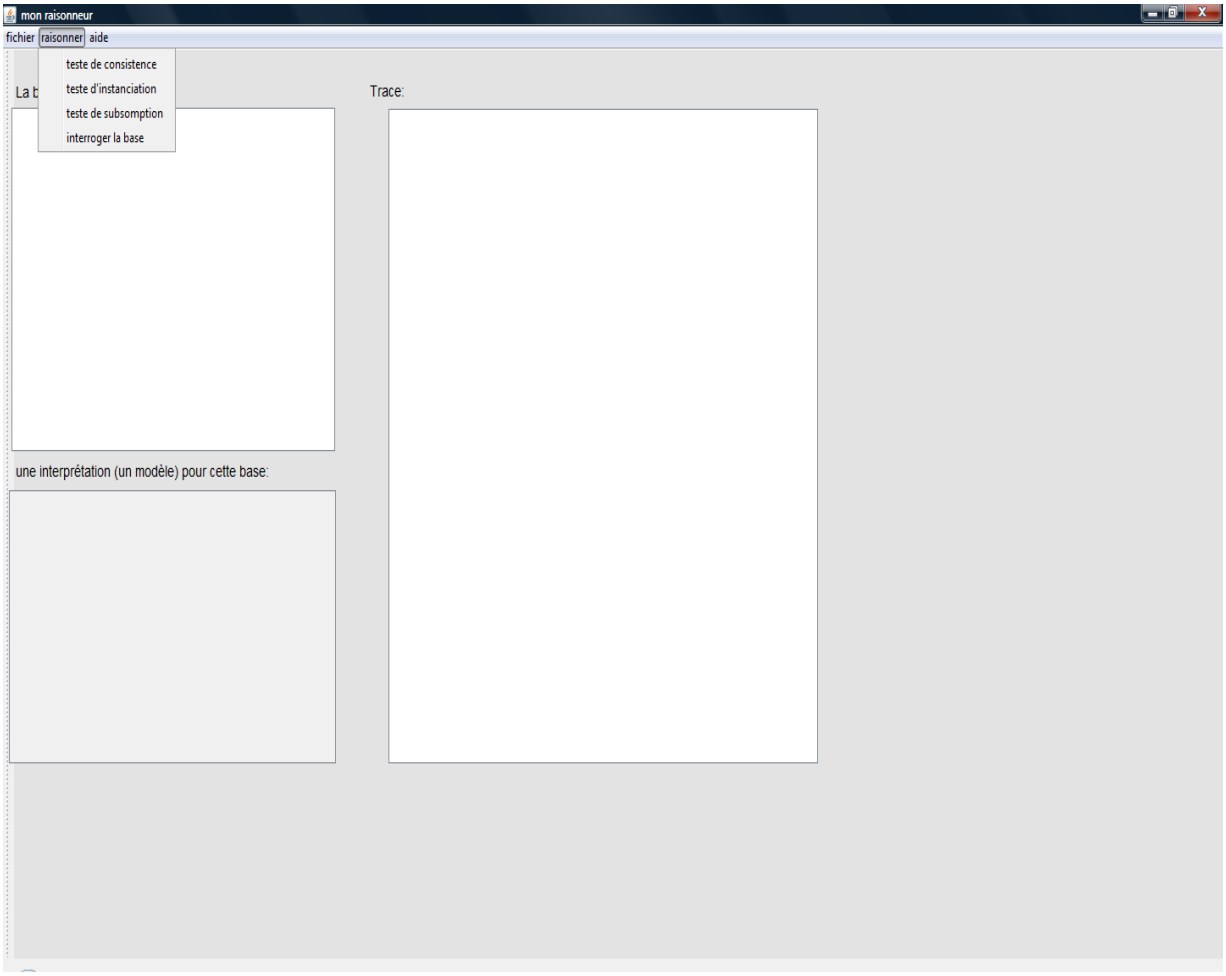

FIG 11. Le sous menu raisonner

3. Les algorithmes développés :

3.1 L'algorithme de subsomption structurelle :

Plusieurs algorithmes de teste de subsomption sont incapables de déterminer la relation de subsomption par rapport a une base de connaissances. Au lieu de cela, ils restreignent le genre des axiomes qui peuvent apparaitre dans la base à des *substitutions d'élimination de dépendance* (*unfolding).* Ces restrictions exigent que tout axiome soit une définition acyclique et elle est unique. Une base satisfait ces restrictions est appelé une base unfoldable.

3.1.1 La technique unfolding :

Soit T une base unfoldable, on veut tester la satisfiabilité d'un concept C, il est possible d'éliminer de C tout nom de concept se produit dans T en utilisant la procédure de substitution récursive appelé *unfolding* **[FRA 03]**. La satisfiabilité du concept résultant est indépendante des axiomes de T et peut être testé en utilisant la procédure de décision qui est capable de déterminer la satisfiabilité d'un seul concept

(i.e. par rapport a un base vide).

Unfold (C, T) $\rightarrow$

Pour tout nom de concept A dans T défini par A ≡ D:

- Remplacer A par D où qu'elle se produit dans C.
- Unfold (D, T).

Pour tout nom de concept A dans T défini par A ⊑ D:

- Remplacer A par D ⊓ A' où qu'elle se produit dans C, (ou A' est un concept non défini dans T ou C).
- Unfold (D, T).

Le concept A' représente ici la caractéristique qui diffère A et D.

La procédure de décision qui trouve l'interprétation de satisfiabilité *I* d'un concept unfoldable peut être maintenant utilisée.

En utilisant la même technique, le raisonnement de subsomption faite aussi indépendamment de T.

Soit deux concept C et D :

T ⊨ C ⊑ D $\Leftrightarrow$ Ø ⊨ Unfold (C, T) ⊑ Unfold (D, T).

Les axiomes de T doivent être des définitions acycliques et uniques.

Si T contient un axiome a définition multiple A, {A ≡ C, A ≡ D} ∈ T, la substitution de A ne préserve pas le sens des deux axiomes. Si T contient des axiomes cycliques, par exemple (A ⊑ ∃ R.A) ∈ T, Unfold (A, T) conduit a une boucle infinie. Si T contient des axiomes généraux, par exemple (∃ R.C ⊑ D) ∈ T, alors on ne peut pas garantir que l'interprétation qui satisfait un concept unfoldable satisfait aussi ces axiomes.

3.1.2 L'algorithme :

Notre algorithme de teste de subsomption (C ⊑ D) est comme suite :

- Unfold (C, T)
- Unfold (D, T)
- Ø ⊨ Unfold (C, T) ⊑ Unfold (D, T), en utilisant l'algorithme NC (chapitre 3, section 5.1)

Soit la terminologie de la figure 5 (utilise *ALCN*), le teste de Woman ⊑GrandMother :

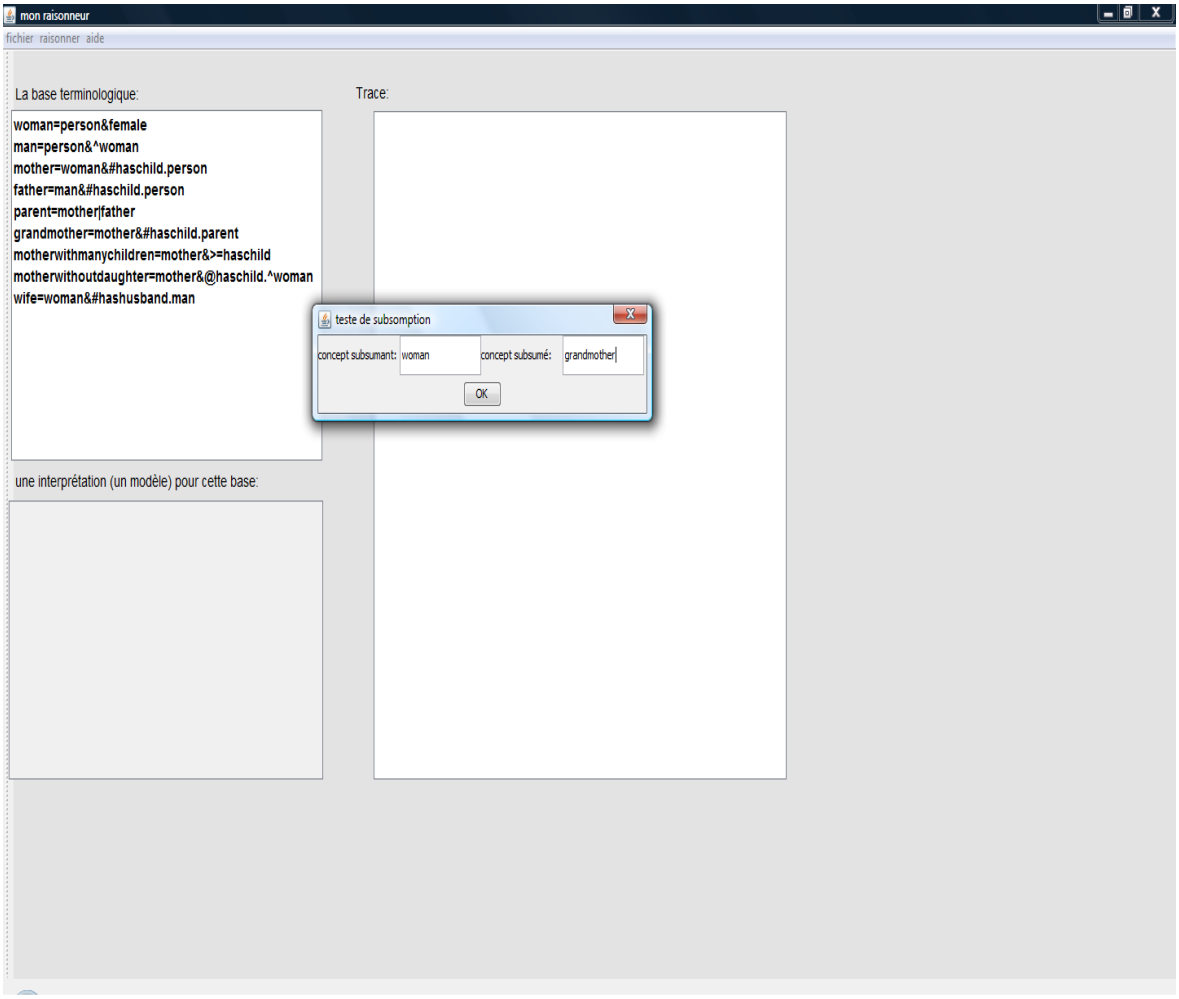

FIG 12. Teste de subsomption.

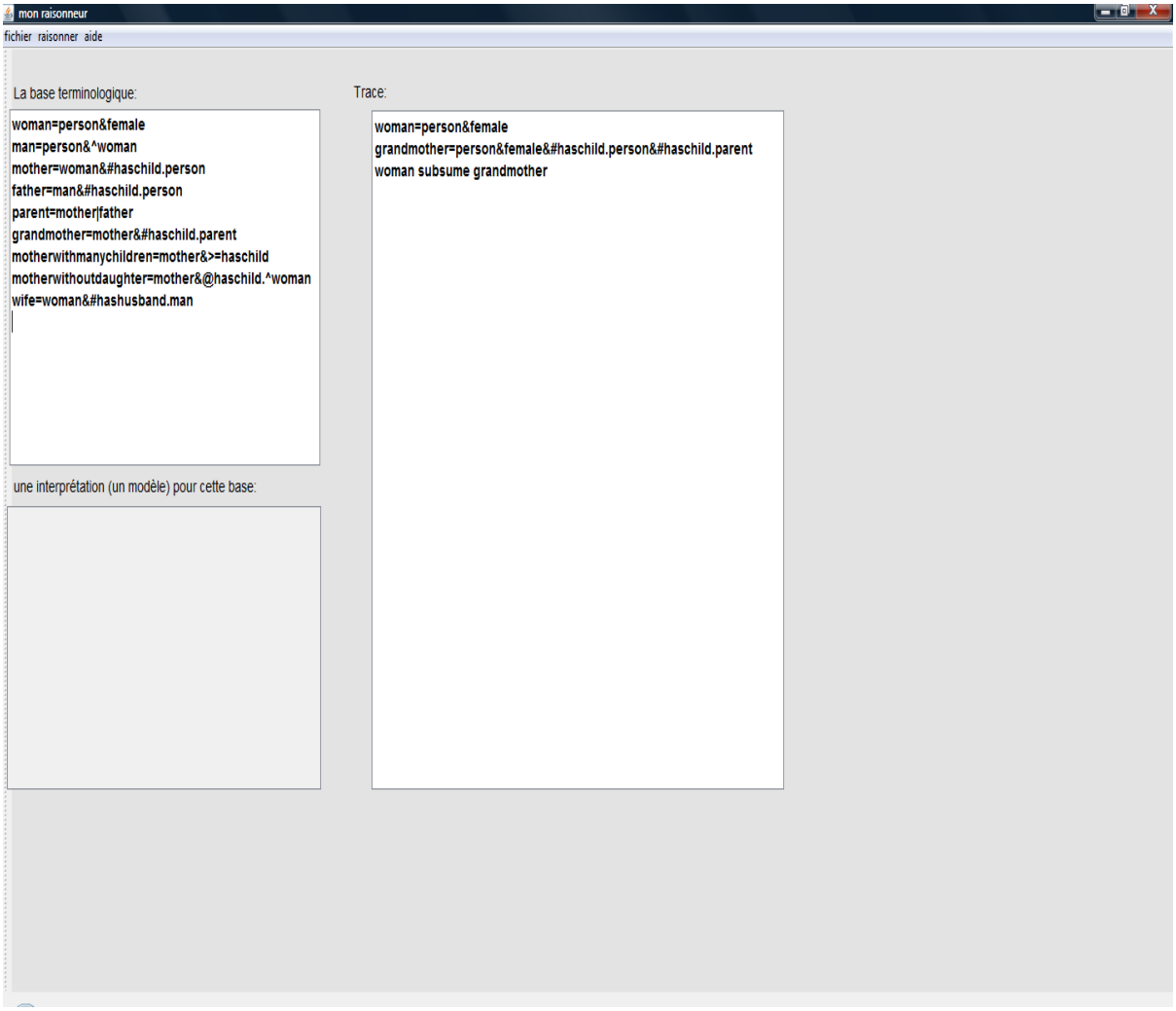

FIG 13. Résultat de teste de subsomption.

Si nous voulons tester si Parent ⊑ Person, nous ne pouvons pas appliquer l'algorithme NC, parce que s'il est facile de vérifier la correction de ces algorithmes, il est plus difficile de prouver qu'un tel algorithme est complet, autrement dit que toutes les relations de subsomption valides sont détectées par l'algorithme. Il s'avère qu'en dehors des langages simples comme *FL-*, la plupart des algorithmes NC sont incomplets, ce qui se montre en exhibant des contre-exemples **[HEI 94]**.

3.1.3 Subsomption et propriétés de treillis :

Les constructeurs et obéissent au règles suivantes **[ROY 93]** :

$A \sqcap A \equiv A$ et $A \mathbf{U} A \equiv A$ (idempotence).

$A \sqcap B \equiv B \sqcap A$ et $A \mathbf{U} B \equiv B \mathbf{U} A$ (commutativité)

$A \sqcap (B \sqcap C) \equiv (A \sqcap B) \sqcap C$ et $A \mathbf{U} (B \mathbf{U} C) \equiv (A \mathbf{U} B) \mathbf{U} C$ (associativité)

Si $D \sqsubseteq C$ et $D \sqsubseteq E$, alors $D \sqsubseteq C \sqcap E$.

Si D ⊑ C et E ⊑ C, alors D **U** E ⊑ C.

Si D ⊑ C, alors D ⊓ X ⊑C, ou X est une description quelconque.

Si D ⊑ C, alors D ⊑C **U** X, ou X est une description quelconque.

L'ensemble des descriptions (potentielles) pour *ALCN* est un treillis, ou chaque couple de concepts (A, B) admet une borne supérieure A **U** B est une borne inférieure A ⊓ B.

Nous pouvons optimiser notre algorithme pour qu'il soit capable de détecter la vérité de la relation de subsomption Parent ⊑ Person :

Nous avons Mother ⊑ Person et Father ⊑ Person alors Mother **U** Father ⊑ Person

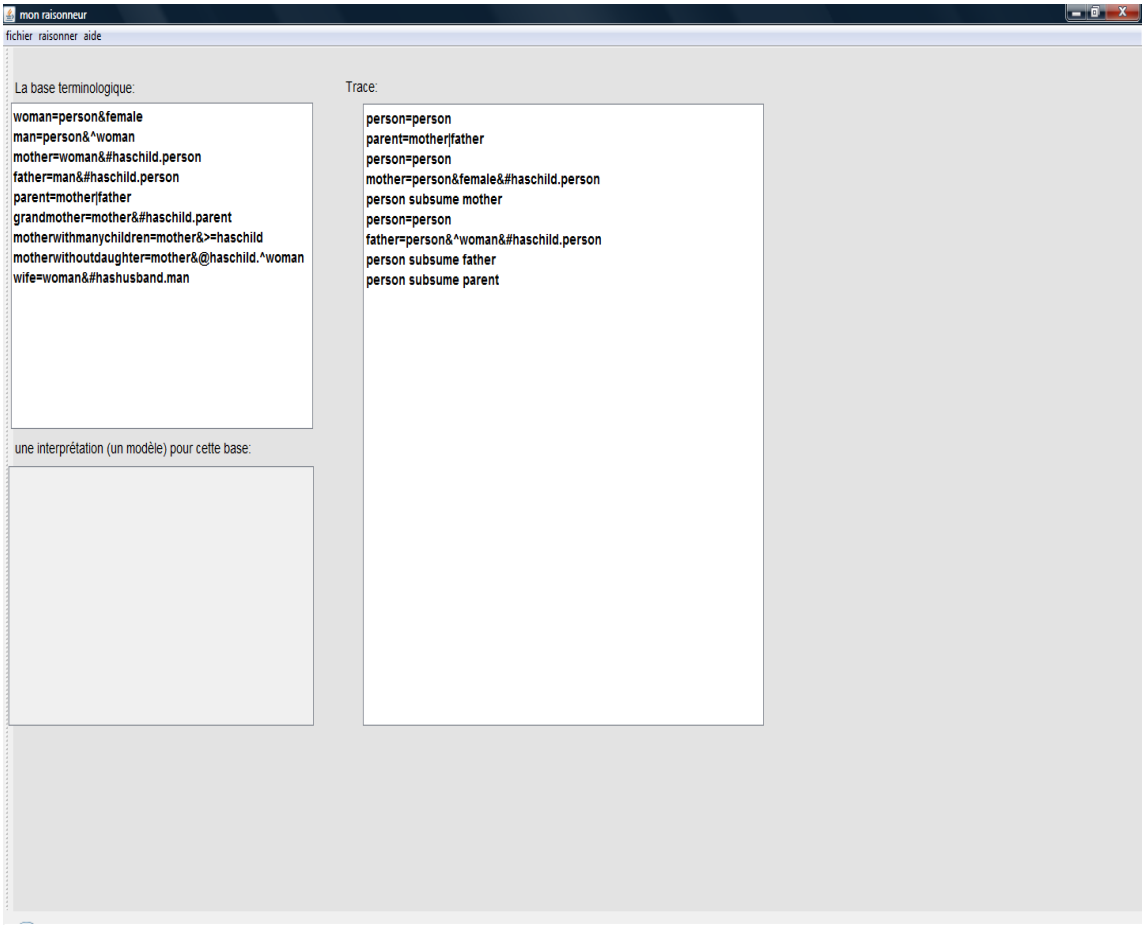

FIG 14. Optimisation de l'algorithme NC.

Cependant, il est plus facile de démontrer la correction et la complétude des algorithmes qui s'appuient sur la méthode des tableaux sémantiques, ainsi d'étudier leur complexité et la décidabilité de la logique associée.

3.2 L'algorithme des tableaux sémantiques :

Considérant la ABox suivante dont il faut tester la consistance:

Pierre enfant Marie

Pierre: $\forall$enfant. $\neg$ MUSICIEN

Marie: $\exists$enfant.HOMME $\sqcap$ $\exists$ami.CYCLISTE

Intuitivement, cette Abox représente trois faits : Marie est l'enfant de Pierre, touts les enfants de Pierre ne sont pas musiciens, Marie a un garçon, et un ami qui est cycliste.

Appliquer la méthode des tableaux sémantiques se fait en trois étapes principales :

1) Transformation de la base en un système de contrainte : qui est un ensemble infini non vide des contraintes de trois formes :

- x : C → x est une instance du concept C.
- x R y → x est en relation avec y par l'intermédiaire du rôle R.
- x $\neq$ y → exprime que les individus x et y ont forcément une interprétation différence.

Les règles de décomposition sont appliquées jusqu'à ce que le système de contrainte devienne complet et plus aucune règle n'est alors applicable. Deux cas peuvent se produire : soit le système est contradictoire et la base associée n'est pas consistante, soit le système est non contradictoire, la base est alors consistante et un modèle de T peut être construit.

Nous avons développé l'algorithme des tableaux sémantiques décrit dans le chapitre 3 section 5.2 pour le langage *ALCN*. Et nous allons montrer formellement que cette ABox est consistante.

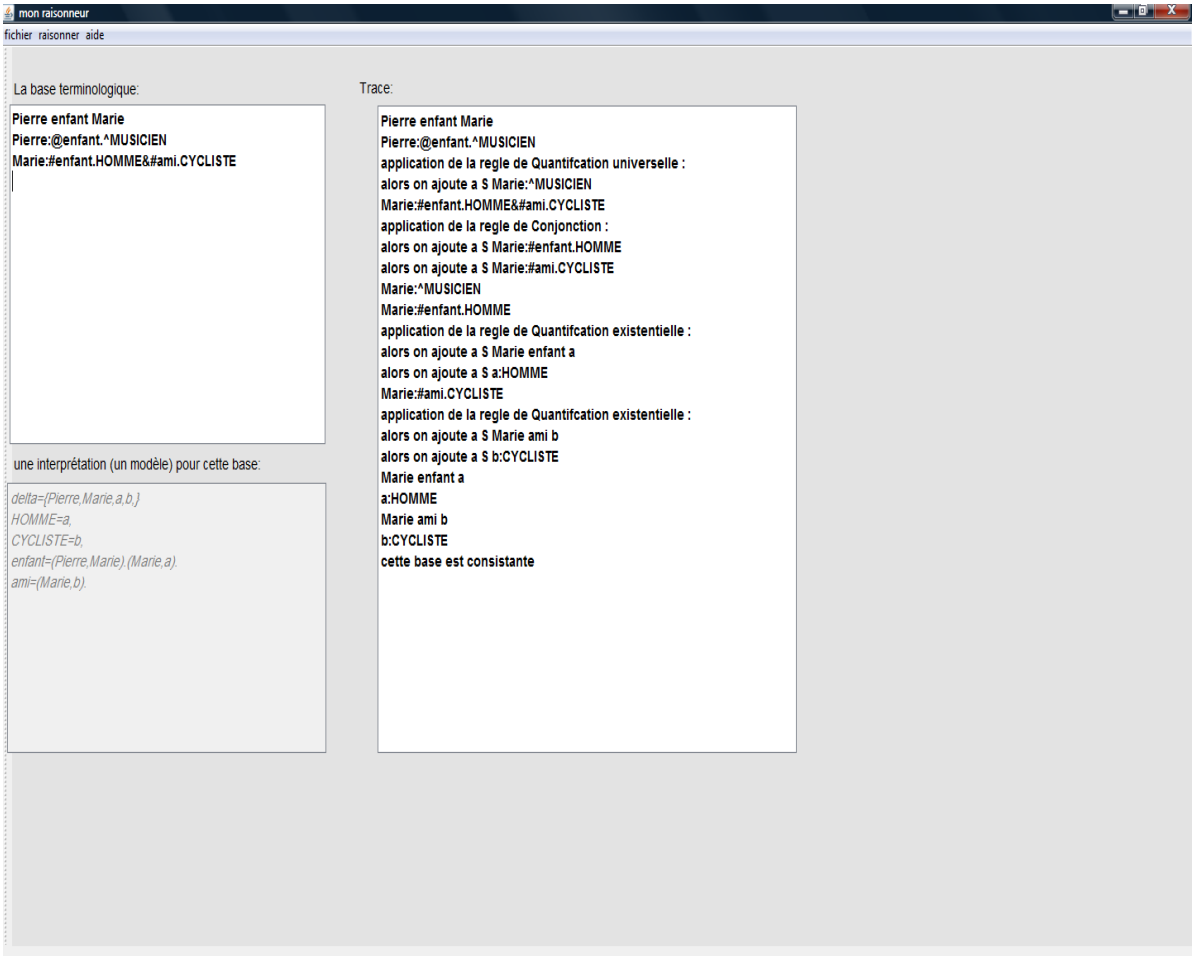

FIG 15. Le teste de consistance avec l'algorithme des tableaux.

Si on ajoute à la ABox précédente le fait : Marie : Musicien, elle devient non consistante.

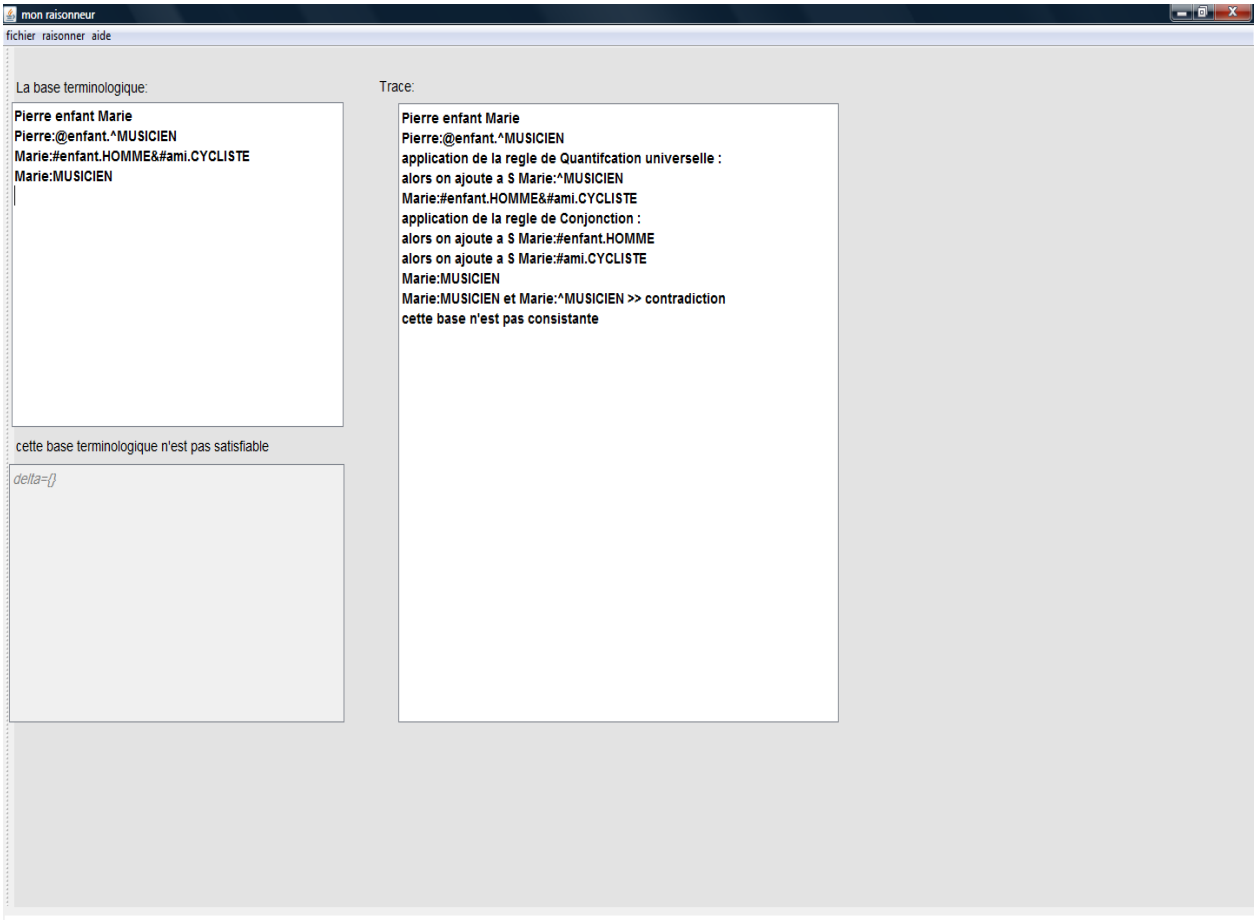

FIG 16. Démonstration du non consistance.

3.3 Variante sur l'algorithme de tableaux sémantiques :

Maintenant nous avons travaillé avec une base qui contient du TBox (des axiomes) et du ABox (des faits).

Dans le cas ou on a les axiomes de type $D \sqsubseteq C$, les contraintes peuvent avoir quatre forme : x : C, x R y, $x \neq y$, et $\forall x$ x: C. Intuitivement, la contrainte $\forall x$ x: C exprime que tout objet x est une instance du concept C.

Cette contrainte permet de traiter une inclusion comme $D \sqsubseteq C$, qui se transforme alors en $\forall x$, x: $\neg D \ \mathbf{U} \ C$.

Au six règle de décomposition du section 5.2 (chapitre 3) doit être ajouté une septième règle:

Règle- $\forall_x \rightarrow$

Si $\forall x$ x: C est dans S (soit S est le système de contrainte).

z est dans S.

z: C n'est pas dans S.

Alors S=S ∪ {z: C}.
Le reste de développement demeure inchangé.

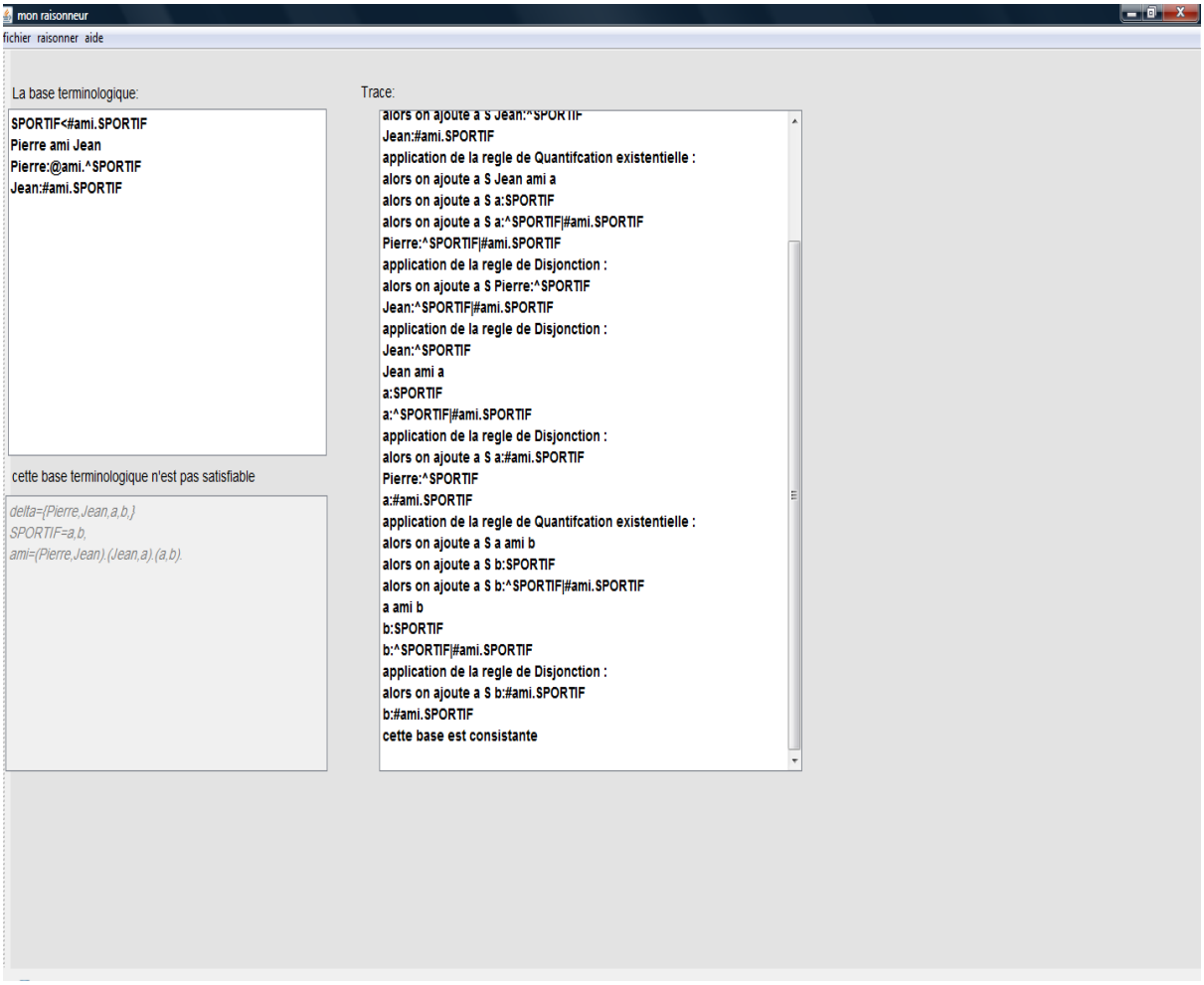

FIG 17. Teste de consistence d'une Abox par rapport à une TBox.

4. Exemples d'interrogations de la base :

Après le teste de consistante de la base et la construction du modèle, il nous est possible d'effectuer des requêtes. Un premier exemple de requête pourrait être le suivant: Quels sont les amis de Pierre ?

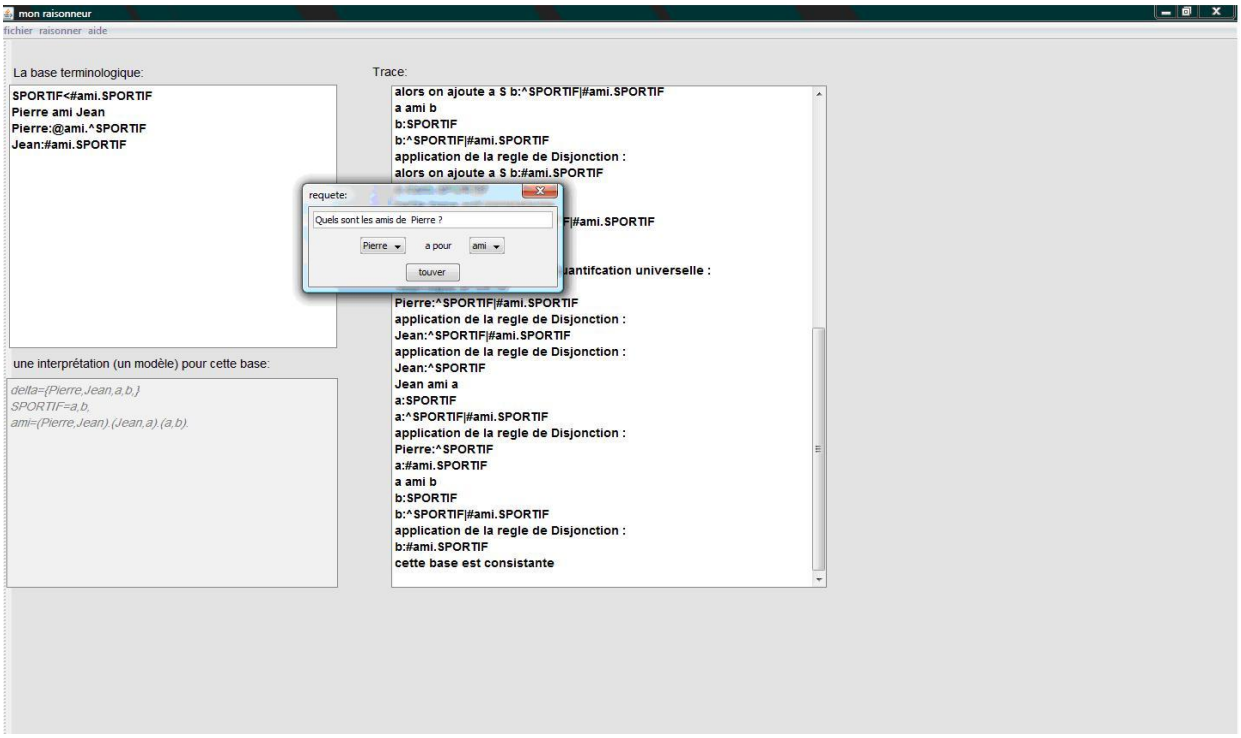

FIG 18. Exemple d'interrogation de la base.

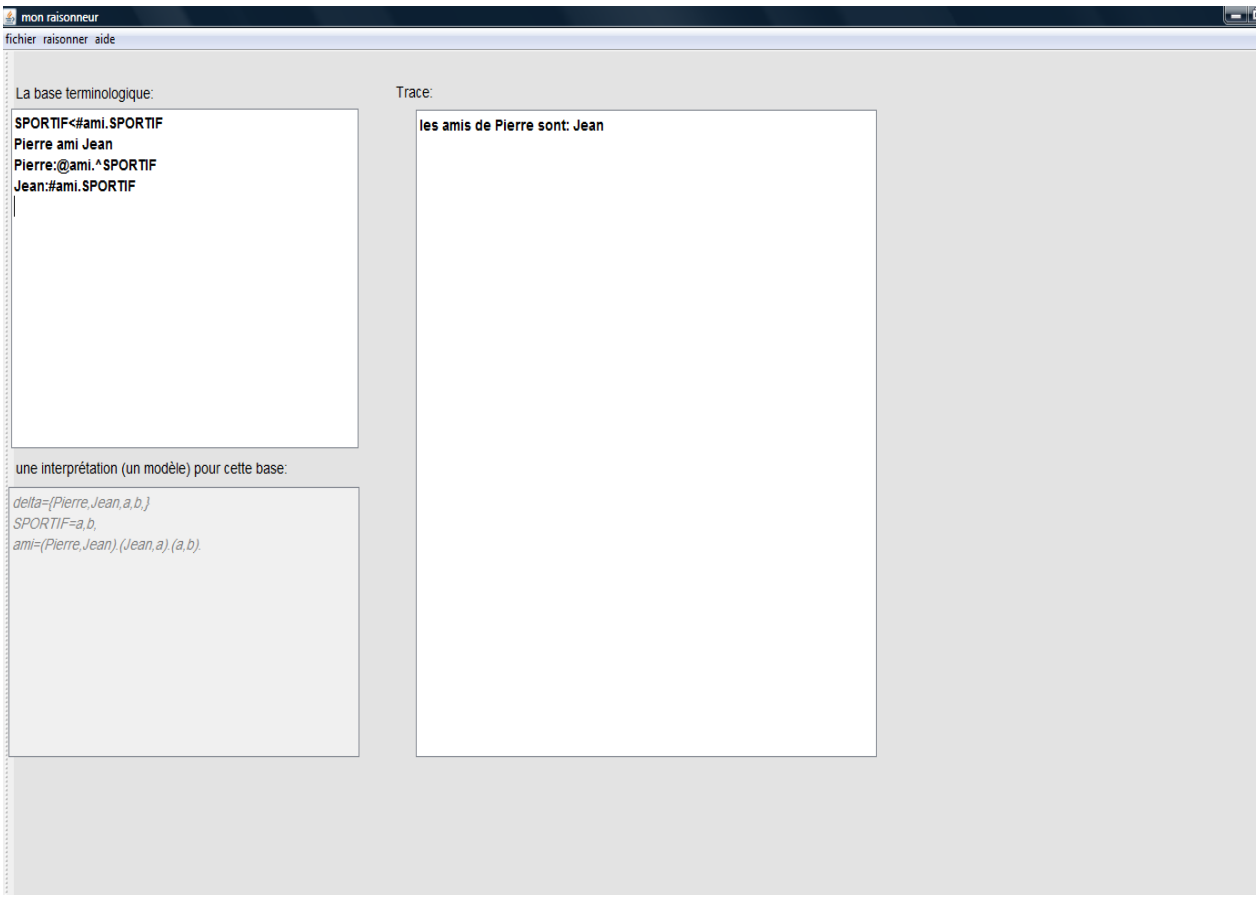

FIG 19. Résultat d'interrogation de la base

Nous pouvons également faire un teste d'instanciation pour connaitre si un individu donné est une instance d'un concept donné.

Par exemple pour la ABox de la figure 15, si on veut tester si Marie est une instance du concept MUSICIEN, on ajout a cette ABox l'assertion Marie : MUSICIEN et on teste si la nouvelle ABox est consistante, si oui alors le teste d'instanciation est positif, si non alors le teste est négatif.

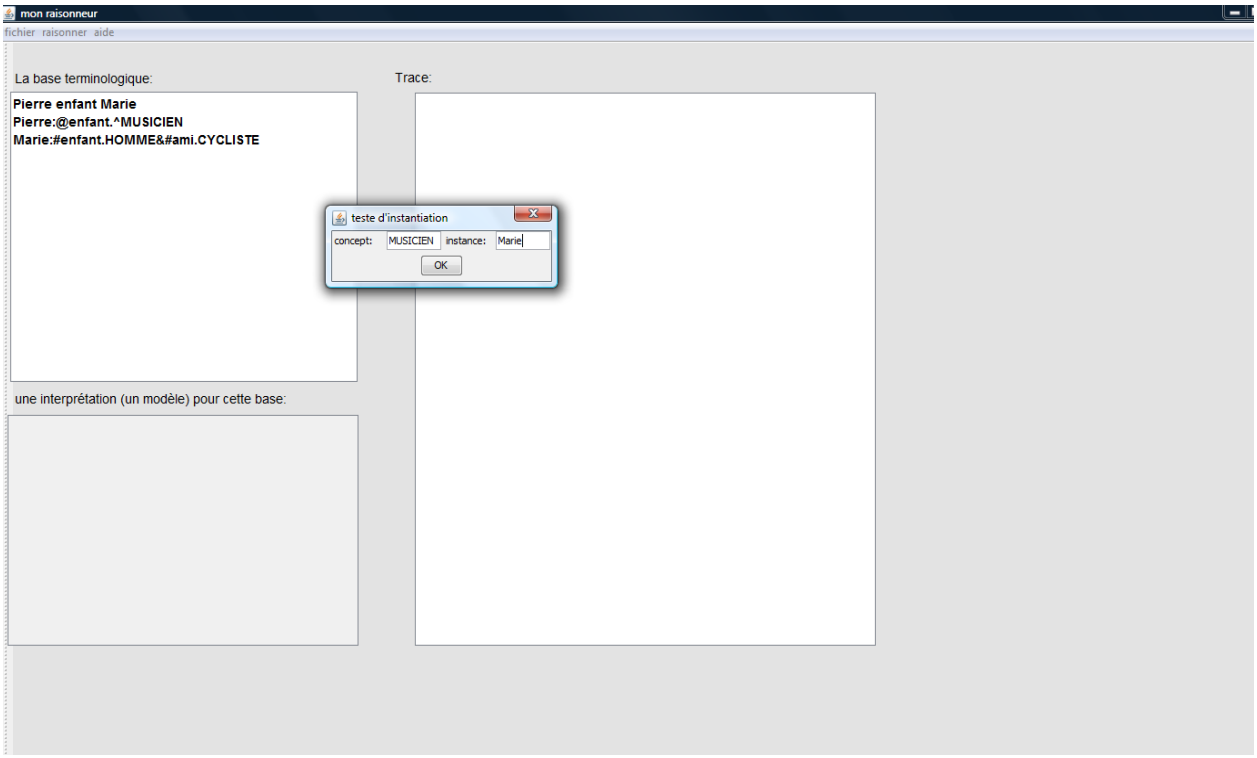

FIG 20. Teste d'instanciation

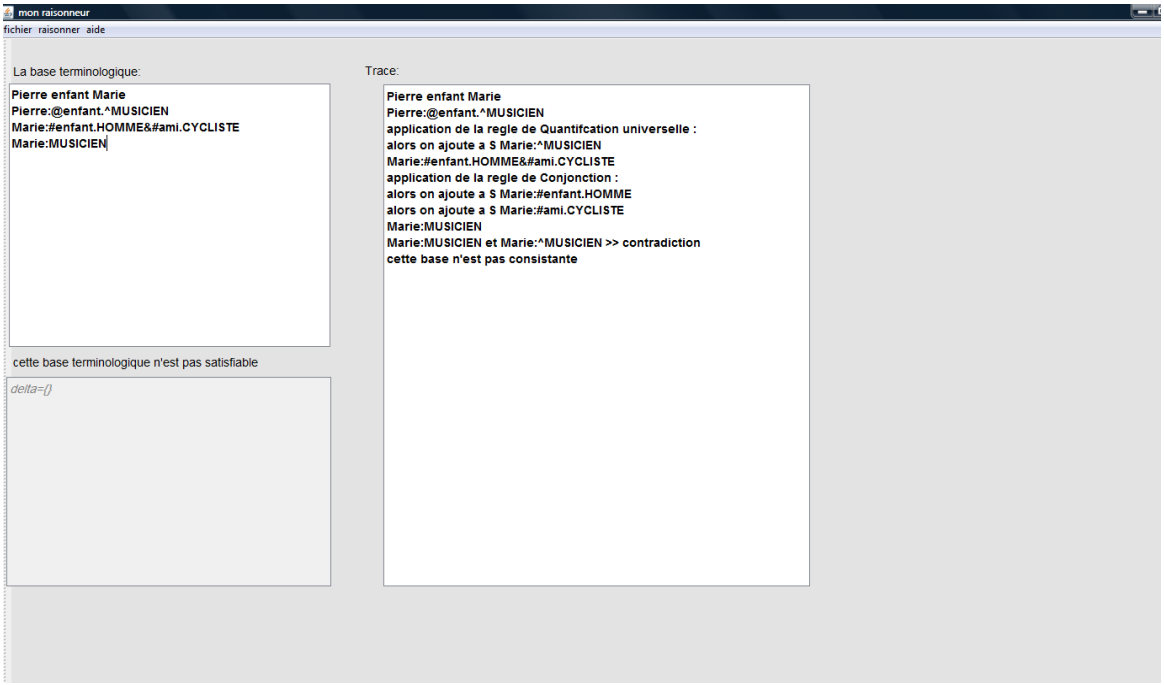

FIG 21. Résultat du teste d'instanciation

La nouvelle ABox n'est pas consistante alors Marie n'est pas une instance du concept MUSICIEN.

5. Conclusion :

Dans ce chapitre, nous avons abordé le raisonnement au sein des logiques de descriptions, pour cela nous avons choisi un langage simple qui est *ALCN* pour exposé les deux voix de traitement de subsomption qui est la relation fondamentale existant entre les descriptions. La méthode structurelle, et malgré sa simplicité, l'inconvénient le plus important de ce type d'algorithme est que, s'il est généralement facile de démontrer la correction de ces règles d'inférence (jamais déduire une relation de subsomption invalide), ils sont généralement incomplets (ils peuvent ne parviennent pas à déduire toutes les relations de subsomption valide). Ainsi il est difficile d'étendre les algorithmes structurels pour faire face aux logiques plus expressives. Contrairement les méthodes des tableaux sémantiques ont une base théorique solide, et elles peuvent être facilement adaptées pour permettent une gamme de langages en changeant l'ensemble des règles de décomposition, et en utilisant des mécanismes de contrôle plus sophistiqués pour assurer la terminaison.

Le chapitre suivant conclut notre travail et permet de dresser des perspectives de recherche.

Conclusion générale et perspectives

« Ce n'est pas la fin.
Ce n'est même pas le commencement de la fin.
Mais, c'est peut-être la fin du commencement »

Winston Churchill

Dans notre thèse nous avons abordé le domaine des logiques de descriptions qui constituent un formalisme de représentation de connaissances caractérisé par les points suivants :

- ✓ Un langage permet de construire des descriptions conceptuelles qui sont génériques (concepts primitifs et définis) ou individuelles (instances).
- ✓ Une sémantique est associée à chaque construction syntaxique par l'intermédiaire d'une interprétation.
- ✓ Une relation de subsomption permet d'organiser les descriptions par niveau de généralité, et de procéder à des inférences ; cette relation est à la base des processus de classification et d'instanciation.

Dans notre contribution nous avons développé une application capable de raisonner avec une base de connaissances modélisées avec les DL, cette modélisation se réalise en deux niveaux. Le premier, le niveau terminologique ou TBox, décrit les connaissances générales d'un domaine alors que le second, le niveau factuel ou ABox, décrit les individus en les nommant et en spécifiant en terme de concepts et de rôles, des assertions qui portent sur ces individus nommés.

Premièrement nous avons développé l'algorithme structurel NC qui s'appuie sur un processus de normalisation à l'issue duquel tous les composants d'une description sont développés et factorisés : toute l'information partagée est recopiée dans le concept traité ; les concepts définis sont remplacés par leur définition, et ainsi tous les symboles d'une définition développée dénotent des concepts ou des rôles primitifs. Le processus de normalisation produit les formes normales des descriptions, qui sont ensuite effectivement comparées. Toutefois, seules les formes normales des concepts définis doivent être comparées, car les relations de subsomption sont systématiquement spécifiées pour les concepts primitifs. Il existe une certaine dualité entre normalisation et comparaison : plus il y a de travail fait lors de la normalisation, moins il y en aura à faire lors de la comparaison. Malgré sa simplicité, cette méthode

prouve ces limites qui sont l'incomplétude et la difficulté de l'étendre pour raisonner sur des logiques plus expressives.

Dans la méthode des tableaux sémantiques, la question « est-ce que C subsume D » est transformée en la question « est-ce que D ⊓¬C est non satisfiable », ce qui suppose que le langage de description des concepts est muni de la négation des concepts définis puis la méthode des tableaux sémantiques est utilisée pour répondre à la dernière question. Il est plus facile de démontrer la correction et la complétude des algorithmes qui s'appuient sur la méthode des tableaux sémantiques, ainsi que d'étudier leur complexité et la décidabilité de la logique associée.

La complexité du raisonnement terminologique et plus spécialement la complexité de la subsomption ont fait couler beaucoup d'encre et ont été des moteurs de la recherche sur les logiques de descriptions. Les travaux sur la complexité du raisonnement terminologique ont mis en évidence que plus un langage de description de concepts et de rôles est riche, plus la complexité du raisonnement est élevée. Le dilemme est alors le suivant : *faut-il préférer un langage limité et concis, qui ne permet pas de tout représenter, mais qui a un comportement déductif contrôlable, ou bien faut-il préférer un langage riche, aux possibilités multiples et variées, mais au comportement déductif imprévisible?*

Les logiques de descriptions ne sont pas seulement un formalisme théorique et réservé aux théoriciens de la représentation des connaissances : la recherche autour des logiques de descriptions est très active et a des visées à la fois pratiques et théoriques. Ainsi, la construction de systèmes traitant de problèmes réels est au centre des préoccupations de nombreux travaux de recherches. Les logiques de descriptions ne sont pas des formalismes figés et sont suffisamment souples pour accepter l'introduction de nouveaux constructeurs, capable de répondre à des besoins particuliers, charge alors au concepteur d'évaluer et de tenir compte de la complexité de la subsomption qui en résulte.

À l'heure actuelle, les conséquences des travaux sur la complexité de la subsomption semblent avoir été assimilées et les réflexions se portent plutôt sur l'utilisation des logiques de descriptions dans des applications et domaines divers comme les bases de données et les systèmes d'information, la fouille de données, le traitement du langage naturel, etc.

Malgré ces points forts, et dans le cadre de la conception de systèmes d'intelligence artificielle réalistes. Les limitations des logiques de descriptions sont de plusieurs sortes, parmi lesquelles figurent :

- la difficulté de représenter des relations entre rôles, comme dans les exemples suivants : *deux personnes qui ont pour résidence des pays différents, ou bien deux enfants d'une même famille qui vont dans deux écoles différentes qui sont éloignées l'une de l'autre de 1 kilomètre* (ces exemples sont tirés de [**SCH 92]**, sur les interactions entre rôles.
- la représentation de quantifications sur les relations comme dans *tous les élèves assistent à tous les cours* **[WOO 91]** ;
- les traitements numériques en général ;
- les définitions récursives ;
- les relations n-aires.

Bien que par l'application de certaines techniques d'optimisation comme l'absorption, le branchement sémantique, la recherche heuristique, … il est possible d'obtenir un système DL basé tableaux sémantiques fonctionne raisonnablement bien dans les cas typiques, et comparable favorablement avec les algorithmes structurels, nous espérons qu'on va arriver a :

- étendre le langage *ALC* pour voir des DL plus expressives, en développons ces algorithmes de subsomption, et étudier leur complexité.
- étudier les différentes techniques d'optimisation appliquées aux algorithmes de tableaux pour les DL expressives et essayer de les implémenter.
- Voir la possibilité d'intégré notre raisonneur dans des applications qui supportent les systèmes de représentation de connaissances.
- continuer avec un esprit de travail collectif.

Références :

[BAC 99] BACHIMONT B., *L'intelligence artificielle comme écriture dynamique : de la raison graphique à la raison computationnelle*, Grasset, Paris, 1999.

[BAC 00] BACHIMONT B., Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances, in *Ingénierie des connaissances : évolutions récentes et nouveaux défis*, Eyrolles, pages 305-323, 2000.

[BER 08] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter F. Patel-Schneider, and Ulrike Sattler. Owl 2: The next step for owl. J. Web Sem., 6(4):309-322, 2008
.

[BOU 95] BOUAUD J., BACHIMONT B., CHARLET J. & ZWEIGENBAUM P, Methodological Principles for structuring an ontology, in *Proceedings of IJCAI'95 Workshop : Basic Ontological Issues in Knowledge sharing*, 1995.

[BOR 94] Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC Description Logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994

[BRA 77] Ronald J. Brachman. What'sin a concept: Structural foundations for semantic networks. *Int. Journal of Man-Machine Studies*, 9(2):127–152,1977.

[BRA 78] Ronald J. Brachman. Structured inheritance networks. In W. A. Woods and R. J. Brachman, editors, *Research in Natural Language Understanding*, Quarterly Progress Report No. 1, BBN Report No. 3742, pages 36–78. Bolt, Beranek and Newman Inc., Cambridge, MA, 1978

[**BRA 85]** Ronald J. Brachman and JamesG. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[CHA 98] CHANDRASEKARAN B., JOSEPHSON J. & BENJAMINS V., The Ontology of Tasks and Methods, in *Proceedings of the 11th workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, 1998.

[DEC 00] DECHILLY T. & BACHIMONT B., Une ontologie pour éditer des schémas de description audiovisuels, extension pour l'inférence sur les descriptions, in *Actes des journées francophones d'Ingénierie des Connaissances (IC'2000)*, 2000.

[DON 91] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. On Artificial Intelligence (IJCAI'91)*, pages458–463, Sydney (Australia), 1991.

[DON 97] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[DOU 05] Doumi N. et A. Lehireche, Une ontologie pour le lexique arabe, in proceeding du 2ème congrès international de « l'ingénierie de la langue arabe et l'ingénierie de la langue », CRSTDLA, UA, 2005.

[FER 97] FERNANDEZ M., GOMEZ-PEREZ A. & JURISTO N., METHONTOLOGY : from ontological art towards ontological engineering, in *Proceedings of the Spring Symposium Series on Ontological Engineering (AAAI'97)*, AAAI Press , 1997.

[FUR 04] Frédéric FÜRST : Contribution à l'ingénierie des ontologies : une méthode et un outil d'opérationalisation, 2004.

[FRA 03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003.*

[GOC 91] P. Gochet and P. Gribomont. Logique, volume 1. Hermès, Paris, 1991.

[GOM 96] GOMEZ-PEREZ A., FERNANDEZ M. & DE VICENTE A. J., Towards a Method to Conceptualize Domain Ontologies, in *Proceedings of the European Conference on Artificial Intelligence, ECAI'96*, pages 41-52, 1996.

[GRU 93] GRUBER T., A translation approach to portable ontology specifications, *Knowledge Acquisition* 5(2), pages 199-220, 1993.

[GUA 94A] GUARINO N., The ontological level, in R. CASATI B. S.&WHITE G., eds., *Philosophy and the cognitive sciences*, Hölder-Pichler-Tempsky, 1994.

[GUA 94B] GUARINO N., CARRARA C. & GIARETTA P., An ontology of meta-level categories, in J. DOYLE F. S.&TORANO P., eds., *Principles of Knowledge representation and Knowledge Reasonning*, Morgan-Kauffmann, pages 270-280, 1994.

[GUA 95] GUARINO N. & GIARETTA P., Ontologies and knowledge bases, towards a terminological clarification, inMARS N., eds., *Towards very large knowledge bases : knowledge building and knowledge sharing*, IOS Press, pages 25-32, 1995.

[GUA 00] GUARINO N. & WELTY C., A Formal Ontology of Properties, in DIENG R. & CORBY O., eds., *Knowledge Engineering and Knowledge Management : Methods, Models and Tools. International Conference*

[HEI 94] J.Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. *An empirical analysis of terminological representation systemes. Artificial Intelligance, 68(2) : 367-397, 1994.*

[HOR 98] Ian Horrocks. The fact system. In TABLEAUX, pages 307-312, 1998.

[HOR 02] Ian Horrocks. Daml+oil: A reason-able web ontology language. In EDBT, pages 2-13, 2002.

[**KAS 99]** KASSEL G. & PERPETTE S., Cooperative ontology construction needs to carefully articulate terms, notions and objects, in *Proceedings of the International Workshop on Ontology Engineering on the global Information Infrastructure*, 1999.

[KAS 00] KASSEL G., ABEL M., BARRY C., BOULITREAU P., IRASTORZA C.&PERPETTE S., Construction et exploitation d'une ontologie pour la gestion des connaissances d'une équipe de recherche, in *Actes des journées francophones d'Ingénierie des Connaissances (IC'2000)*, 2000.

[KAS 02] KASSEL G., OntoSpec : une méthode de spécification semi-informelle d'ontologies, in *Actes des journées francophones d'Ingénierie des Connaissances (IC'2002)*, pages 75-87, 2002.

[KAY 97] KAYSER D., *La représentation des connaissances*, Hermès, 1997.

[LEH 06] A. Lehireche, A. Rahmoun, ON LINE LEARNING : EVOLVING in REAL TIME a neural Net Controller of 3D-robot-arm. Track and Evolve, 4th ACS/IEEE international conference on Computer Systems and Applications (AICCSA-06),, Dubai/Sharjah, UAE. 2006.

[LEH 10] A. Lehireche, Reda Mohamed Hamou, A. Chaouki Lokbani, Mohamed Rahmani : Representation of textual documents by the approach WordNet and n-grams for the unsupervised classification (clustering) with 2D cellular automata : a comparative study. Computer and Information Science 3(3) : 240-255 (2010).

[LEV 87] Hector J. Levesque and Ron J. Brachman. Expressiveness and tractability in knowledge representation and reasoning *Computational Intelligence*, 3:78–93, 1987.

[MIZ 97] MIZOGUCHI R.&IKEDA M., Towards ontolgy engineering, in *Proceedings of the Joint Pacific Asian Conference on Expert Systems*, 1997.

[NAP 97] A. Napoli. "*Une introduction aux logiques de descriptions*". Rapport de recherche N° 3314. 1997.

[NOB 00] NOBÉCOURT J. & BIÉBOW B., MDOS : a modelling language to build a formal ontology en either Description Logics or Conceptual Graphs, in *Proceedings of the 12th International Conference on Knowledge Engineering and Management (EKAW'2000)*, Springer-Verlag LNAI 1937, pages 57-64, 2000.

[PAT 89] Peter F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39:263–272, 1989.

[ROY 93] V. Royer and J.J Quantz. Deriving inference rules for description logics : A rewriing approch into sequent calculi. Kit-Report 111, technische Universitat, Berlin, 1993.

[SCH 89] Manfred Schmidt-Schau. Subsumption in KL-ONE is undecidable. In Ron J. Brachman, Hector J. Levesque, and Ray Reiter, editors, *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'89)*, pages421–431. Morgan Kaufmann, LosAltos , 1989.

[SCH 91] Manfred Schmidt-Schau and Gert Smolka, Attributive concept descriptions with complements. *Artificial Intelligence*,

[SCH 92] A. Schmiedel. For a more expressive query language. In AAAI Fall Symposium Series - Issues in Description Logics: Users Meets Developers, Cambridge (MA), USA, pages 98_102, 1992.

[SMO 88] Gert Smolka. A feature logic with subsorts. Technical Report 33, IWBS, IBM Deutschland, P.O. Box 80 08 80 D-7000 Stuttgart 80, Germany, 1988. 48(1):1–26, 1991.

[SOW 00] SOWA J., Ontology, Metadata and semiotics, in *Proceedings of the 8th International Conference on Conceptual Structures (ICCS'2000)*, Springer-Verlag, pages 55-81, 2000.

[STA 00] STAAB S. & MAEDCHE A., Axioms are objects too : Ontology Engineering beyong the modeling of concepts and relations, research report 399 of the AIFB Institute (Karlsruhe), 2000.

[TEU 01] TEULIET R. & GIRARD N., Des connaissances pour l'action dans les organisations. Quelle ingénierie des connaissances pour assister l'activité ?, in *Actes des journées francophones d'Ingénierie des Connaissances (IC'2001)*, Presse Universitaire Grenobloise, pages 253-272, 2001.

[TOU 06] Toumouh A, Lehireche A, D Widdows, M. Malki. Adapting WordNet to the Medical Domain using Lexicosyntactic Patterns in the Ohsumed Corpus : 4th ACS/IEEE international conference on Computer Systems and Applications (AICCSA-06),, Dubai/Sharjah, UAE. 2006.

[TSA 03] Tsarkov, D. et Horrocks, I., 2003. DL reasoner vs. rst-order prover. Dans Proc. of the 2003 Description Logic Workshop (DL 2003) volume. pp. 152159.

[USC 95] USCHOLD M. & KING M., Towards a methodology for building ontologies, in *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI'95*, 1995.

[USC 96] USCHOLD M., Building ontologies : towards a unified methodolgy, in *Proceedings of the 16th conference of the British Computer Society Specialist Group on Expert Systems*, 1996.

[WEL 01] WELTY C. & GUARINO N., Supporting ontological analysis of taxonomic relationships, *Data et Knowledge Engineering* (39), pages 51-74, 2001.

[**WOO 91]** W.A.Woods. Understanding subsumption and taxonomy: A framework for progress. In J.F. Sowa, editor, Principles of Semantic Networks: Explorations in the Representation of Knowledge, pages 45_94. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

Note sur les logiques de descriptions et le Web

Les lecteurs navigateurs pourront aller visiter l'adresse suivante sur le Web : http://dl.kr.org/dl

Où ils trouveront des informations concernant notamment la bibliographie et les séminaires annuels sur les logiques de descriptions, ainsi que les moyens de joindre les chercheurs du domaine et d'obtenir les systèmes disponibles