

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES  
DE SOPHIA ANTIPOLIS  
UMR 6070

# RAISONNEMENT SUR DES ONTOLOGIES DÉCOMPOSÉES

*Thi Anh Le PHAM, Nhan LE THANH*

*Projet MAINLINE*

Rapport de recherche  
ISRN I3S/RR–2006-15–FR

Mai 2006

---

RÉSUMÉ :

Le raisonnement dans une base de connaissance (BC) est une des applications les plus importantes des raisonneurs de la logique de description (LD). Les exigences de l'exécution du temps et de l'espace de mémoire sont les deux facteurs significatifs qui influencent directement la performance d'un algorithme de raisonnement. Dans ce rapport, nous étudions une nouvelle technique pour optimiser le raisonnement de LD avec le but de réduire deux facteurs ci-dessus au minimum possible. Cette technique est appliquée pour accélérer le raisonnement de TBox et d'ABox, particulièrement de grands TBoxes. L'incorporation de cette technique avec des techniques d'optimisation précédentes dans les systèmes courants de DL peut efficacement résoudre des inférences insurmontables. Notre technique s'appelle la décomposition de l'ontologie où la décomposition d'une ontologie en plusieurs sous-ontologies est appliquée telle qu'elle préserve toujours la sémantique et les services d'inférence de l'ontologie originale. Nous nous intéressons au fait que comment raisonner efficacement avec les BCs multiples et améliorer l'efficacité du raisonnement sur des ontologies composantes.

MOTS CLÉS :

optimisation, logiques de description distribuées, raisonnement, décomposition d'ontologie.

---

ABSTRACT:

Reasoning in a Knowledge Base (KB) is one of the most important applications of Description Logic (DL) reasoners. The execution time and storage space requirement are both significant factors that directly influence the performance of a reasoning algorithm. In this paper, we investigate a new technique for optimising DL reasoning with the purpose to minimize two factors above as much as possible. This technique is applied to speed-up TBox and ABox reasoning, especially for large TBoxes. The incorporation of this technique with previous optimisation techniques in current DL systems can effectively solve intractable inferences. Our technique is called "ontology decomposing" in which decomposition of one ontology to many sub-ontologies is implemented such that it still preserves the semantic and inference services of original ontology. We are concerned about how to reason effectively with multiple KBs and how to improve the efficiency of reasoning over component ontologies.

KEY WORDS :

optimisation, distributed description logics, reasoning, ontology decomposing

# Raisonnement sur des Ontologies décomposées

Thi Anh Le PHAM, Nhan LE-THANH

Laboratoire I3S, UMR 6070, CNRS  
Université de Nice Sophia-Antipolis, France  
tpham@i3s.unice.fr, Nhan.Le-Thanh@unice.fr

## Abstract

Reasoning in a Knowledge Base (KB) is one of the most important applications of Description Logic (DL) reasoners. The execution time and storage space requirement are both significant factors that directly influence the performance of a reasoning algorithm. In this paper, we investigate a new technique for optimising DL reasoning with the purpose to minimize two factors above as much as possible. This technique is applied to speed-up TBox and ABox reasoning, especially for large TBoxes. The incorporation of this technique with previous optimisation techniques in current DL systems can effectively solve intractable inferences. Our technique is called "ontology decomposing" in which decomposition of one ontology to many sub-ontologies is implemented such that it still preserves the semantic and inference services of original ontology. We are concerned about how to reason effectively with multiple KBs and how to improve the efficiency of reasoning over component ontologies.

**Key words:** optimisation, distributed description logics, reasoning, ontology decomposing.

## Résumé en français

Le raisonnement dans une base de connaissance (BC) est une des applications les plus importantes des raisonneurs de la logique de description (LD). Les exigences de l'exécution du temps et de l'espace de mémoire sont les deux facteurs significatifs qui influencent directement la performance d'un algorithme de raisonnement. Dans ce rapport, nous étudions une nouvelle technique pour optimiser le raisonnement de LD avec le but de réduire deux facteurs ci-dessus au minimum possible. Cette technique est appliquée pour accélérer le raisonnement de TBox et d'ABox, particulièrement de grands TBoxes. L'incorporation de cette technique avec des techniques d'optimisation précédentes dans les systèmes courants de DL peut efficacement résoudre des inférences insurmontables. Notre technique s'appelle la « décomposition de l'ontologie » où la décomposition d'une ontologie en plusieurs sous-ontologies est appliquée telle qu'elle préserve toujours la sémantique et les services d'inférence de l'ontologie originale. Nous nous intéressons au fait que comment raisonner efficacement avec

les BCs multiples et améliorer l'efficacité du raisonnement sur des ontologies composantes.

**Mots clés** : optimisation, logiques de description distribuées, raisonnement, décomposition d'ontologie.

## 1 Introduction

1. Comme nous savons, la complexité du calcul du raisonnement est un thème important dans la recherche de la logique de description, la plupart des résultats complexes disponibles se concentrent sur le raisonnement avec seulement des concepts et ne tiennent pas compte de TBoxes (acycliques). Des dernières LDs ont utilisés le déploiement ("unfolding" en anglais) pour réduire le raisonnement avec des TBoxes en raisonnement avec des concepts. Le déploiement d'un concept C par rapport à un TBox T signifie remplacer itérativement des noms de concept dans C par leurs définitions données dans T

Dans l'article [6] de Nebel, il a prouvé que, dans le pire cas, le déploiement peut produire une explosion exponentielle de la taille de concept. Puisque la complexité du raisonnement avec des logiques de description n'est habituellement pas ExpSpace-dur, ce résultat prouve que le déploiement n'est pas une manière appropriée de traiter des TBoxes. Nebel a également prouvé que dans la réalité et dans des applications pratiques, le pire cas n'est presque jamais trouvé. En raison de la plupart de ces résultats, l'analyse de la complexité du raisonnement avec des TBoxes a été négligée il y a longtemps.

2. Un caractère essentiel de la logique de description est sa capacité de représenter et de raisonner au sujet de la connaissance terminologie. La connaissance terminologie est stockée dans les prétendus TBoxes ("so-called TBoxes") qui viennent principalement de deux types. Prétendus TBoxes acycliques sont les ensembles des définitions de concept qui peuvent être considérées comme définitions non récurrente tandis que TBoxes généraux permettent à présenter des concepts arbitraires et complexes. Dans ces dernières années, on se concentre essentiellement sur le raisonnement avec des TBoxes acycliques (TA) car:

- Les TBoxes généraux (TG) subsument les TBoxes acycliques
- Pour plusieurs LDs, le raisonnement avec TA est moins complexe que le raisonnement avec TG.
- Il existe des LDs que le raisonnement avec TG est indécidable cependant que le raisonnement avec TA est décidable.

3. Le raisonnement dans une base de connaissance (BC) est un des applications les plus importantes des raisonneurs de la logique de description (LD). Les exigences du temps d'exécution et de l'espace de mémoire sont les deux facteurs significatifs qui influencent directement la performance d'un algorithme de raisonnement. Pour les grands TBoxes, le raisonnement efficace est un grand défi en raison de les inférences insurmontables. Une cause importante est produite par les Inclusions de Concept Général (ICGs). Chaque ICG cause une expression disjonctive qui est ajoutée à l'étiquette de chaque noeud dans l'arbre produit par l'algorithme de

tableaux. Elle mène à une augmentation exponentielle de la taille de l'espace de recherche pour être explorée par la règle  $\neg\perp$ . En conséquence, un nombre réduit de ICGs peut considérablement dégrader l'exécution d'un raisonneur de LD. Alors des techniques évidentes d'optimisation du raisonnement se concentrent souvent sur la réduction de la plupart des ICGs d'une terminologie. Notre technique, parallélisant des algorithmes de tableaux, n'est pas en dehors de ce but. Au lieu du raisonnement sur une ontologie donnée, les algorithmes de tableaux sont concurremment mis en application sur plusieurs sous-ontologies, qui sont produits de l'ontologie originale. Nous présentons donc la décomposition d'une ontologie en plusieurs tels sous-ontologies dans le même domaine de la première. Notre technique s'appelle la "décomposition overlay" où des ontologies décomposées (appelés sous-ontologies) conservent tous les concepts primitifs, tous les rôles primitifs et tous les concepts définis de l'ontologie originale, seulement l'ensemble des axiomes est divisé en plusieurs ensembles d'axiomes. Par conséquent, le nombre d'axiomes est significativement réduit. Dans ce rapport, l'algorithme de décomposition n'est pas présenté. Nous nous concentrons seulement sur les propriétés qui sont les plus importantes des ontologies décomposées, elles comprennent la conservation de la syntaxe, de la sémantique et d'inférences de l'ontologie originale. Le raisonnement dans le système des ontologies décomposées est mis en application avec deux méthodes: méthode distribuée et méthode parallèle. Dans la première méthode, des ontologies décomposées sont représentés dans un système distribué par la logique de description distribuée (LDD) et un algorithme de tableau distribué est appliqué. Pour la deuxième, un algorithme de raisonnement est l'algorithme de tableau normal. Les algorithmes sont conçus de sorte que les résultats de raisonnement dans des ontologies décomposées soient les mêmes que dans l'ontologie originale.

Ce rapport est organisé comme suit: les sections 2 et 3 discutent des travaux relatifs, nous rappelons plusieurs techniques d'optimisation qui sont appliquées dans les systèmes courants de raisonnement de LD et aussi quelques définitions en LDD. Dans la section 4, nous définissons une décomposition overlay d'un TBox présenté par un TBox distribué et ses propriétés. Les sections 5 et 6 présentent les algorithmes pour le raisonnement parallèle et le raisonnement dans LDD, et nous prouvons également brièvement la solidité et la perfection de l'algorithme de raisonnement dans LDD.

## 2 Des techniques d'optimisation

Le raisonnement dans LDs se base essentiellement sur l'algorithme de tableau. L'algorithme de tableau normal est trop lent pour former une base d'un système utile de logique de description. On a donc étudié et a utilisé un rayon connu d'adaptation et des optimisations du procédé qui améliorent l'exécution de l'algorithme du test de la satisfaisabilité.

Pour poser une optimisation efficace, nous étudions également quelques techniques d'optimisation précédentes qui sont appliqués dans notre cas.

## 2.1 Normalisation lexicale et encodage

La détection des contradictions (clashes) peut être adressée en transformant des expressions de concepts (et leurs sous-expressions) en forme lexicale normalisée, et en identifiant des expressions lexicales équivalentes. Alors toutes les expressions de concepts peuvent être traitées également avec une contradiction qui est détectée lorsqu'une expression de concept et sa négation s'apparaissent dans le même label de noeud. Dans cette forme lexicale normalisée, les expressions de concepts consistent seulement en concepts atomiques, concepts de conjonction, concept universel de rôle, et leurs négations. On peut combiner cette technique avec "lazy unfolding" [2].

*Exemple 1:*  $\exists R.(C \sqcap D) \sqcap \forall R.C$  serait transformé en  $\sqcap \{ \neg(\forall R. \neg \sqcap \{ C, D \}), \forall R. \neg C \}$  une contradiction sera immédiatement détectée, indépendamment de la structure de C.

*Exemple 2:* Normalisation et encodage d'expression  $\exists R.(C \sqcap \neg D \sqcap \neg E) \sqcap \forall R.(\neg C \sqcup D \sqcup E)$

1. Normaliser la première sous-expression:

$$Normalise(\exists R.(C \sqcap \neg D \sqcap \neg E)) \longrightarrow \neg Normalise(\forall R. \neg(C \sqcap \neg D \sqcap \neg E))$$

$$Normalise(\forall R. \neg(C \sqcap \neg D \sqcap \neg E)) \longrightarrow Encode(\forall R. Normalise(\neg(C \sqcap \neg D \sqcap \neg E)))$$

$$Normalise(\neg(C \sqcap \neg D \sqcap \neg E)) \longrightarrow \neg Encode(C \sqcap \neg D \sqcap \neg E)$$

$$Encode(C \sqcap \neg D \sqcap \neg E) \longrightarrow CN_1, \text{ où } CN_1 \text{ est un nouveau nom de concept et } T \longrightarrow T \cup \{CN_1 \equiv (C \sqcap \neg D \sqcap \neg E)\}$$

$$Encode(\forall R. Normalise(\neg(C \sqcap \neg D \sqcap \neg E))) \longrightarrow CN_2, \text{ où } T \longrightarrow T \cup \{CN_2 \equiv \forall R. \neg CN_1\}$$

$$Normalise(\exists R.(C \sqcap \neg D \sqcap \neg E)) \longrightarrow \neg CN_2$$

2. Normaliser la deuxième sous-expression:

$$Normalise(\forall R.(\neg C \sqcup D \sqcup E)) \longrightarrow Encode(\forall R. Normalise(\neg C \sqcup D \sqcup E))$$

$$Normalise(\neg C \sqcup D \sqcup E) \longrightarrow Normalise(\neg(C \sqcap \neg D \sqcap \neg E))$$

$$Normalise(\neg(C \sqcap \neg D \sqcap \neg E)) \longrightarrow \neg Encode(C \sqcap \neg D \sqcap \neg E)$$

$$Encode(C \sqcap \neg D \sqcap \neg E) \longrightarrow CN_1$$

$$Normalise(\forall R.(\neg C \sqcup D \sqcup E)) \longrightarrow Encode(\forall R. \neg CN_1) \longrightarrow CN_2$$

3. Recombiner les deux sous-expressions:

$$Normalise(\exists R.(C \sqcap \neg D \sqcap \neg E) \sqcap \forall R.(\neg C \sqcup D \sqcup E)) \longrightarrow Normalise(\neg CN_2 \sqcap CN_2) \longrightarrow \perp$$

## 2.2 Recherche de l'embranchement sémantique (Semantic Branching Search)

Les algorithmes du standard de tableau sont eux même inefficace parce qu'ils emploient une technique de recherche basée sur l'embranchement syntaxique. En étendant le label d'un noeud  $x$ , l'embranchement syntaxique réalise le choix d'une disjonction non-expansive  $(C_1 \sqcup \dots \sqcup C_n)$  dans  $L(x)$  et la recherche des différents modèles

obtenus en ajoutant chacun de disjuncts  $C_1, \dots, C_n$  à  $L(x)$ . Comme les branches alternatives de l'arbre de recherche ne doivent pas disjoindre, il n'y a rien à empêcher la répétition d'un disjunct insatisfait dans des branches différentes. Ce problème peut mener à un grand gaspillage.

Au lieu de choisir une disjonction non-expansive dans  $L(x)$ , on a choisi un disjunct simple  $D$  d'une disjonction non-expansive dans  $L(x)$ . Les deux sous-arbres qui peuvent être obtenus en ajoutant  $D$  ou  $\neg D$  à  $L(x)$  sont alors recherchés. Puisque les deux sous-arbres sont strictement disjoints, il n'y a aucune possibilité de recherche gaspillée comme dans l'embranchement syntactique.

### 2.3 Simplification

Simplification est une technique employée pour réduire la quantité des indéterministes (l'embranchement) dans l'expansion des labels de noeud. Avant que n'importe quelle expansion indéterminée d'un label de noeud  $L$  soit exécutée, des disjonctions (conjonctions réellement niées) de  $L$  sont examinées, et simplifiées si possible. La simplification doit étendre la manière de déterministe des disjonctions dans  $L(x)$  qui présentent seulement une possibilité d'expansion et pour détecter une contradiction quand une disjonction de  $L$  n'a aucune possibilité d'expansion. Cette simplification s'est appelée la propagation de contrainte booléenne (PCB). En effet, la règle d'inférence

$$\frac{\neg C_1, \dots, \neg C_n, C_1 \sqcup \dots \sqcup C_n \sqcup D}{D}$$

est employée pour simplifier l'expression représentée par  $L(x)$ .

*Exemple 3:*  $\{(C \sqcup (D_1 \sqcap D_2)), (\neg D_1 \sqcup \neg D_2), \neg C\} \subseteq L(x)$

PCB étend la manière de déterministe de la disjonction  $(C \sqcup (D_1 \sqcap D_2))$  parce que  $\neg C \in L(x)$ . L'expansion déterminée de  $(D_1 \sqcap D_2)$  ajoute  $D_1$  et  $D_2$  à  $L(x)$ , permettant à PCB d'identifier  $(\neg D_1 \sqcup \neg D_2)$  comme une contradiction sans occurrence s'embranchante.

### 2.4 Absorption de ICG

Cette technique vise à réduire le nombre de ICGs en les absorbant en des axiomes d'introduction de concept primitif autant que possible. La structure de ICGs absorbables est proposée par la syntaxe du langage de description de concept de Grail dans la terminologie de Galen, ils appartiennent à un des formes suivantes:

- GCI dont l'antécédent est seulement un nom primitif de concept.
- GCI dont l'antécédent est soit une expression conjonctive de concept soit un nom de concept non-primitif dont la définition est une expression conjonctive de concept.

Dans ce contexte, la première proposition conjointe d'une expression conjonctive de concept est toujours soit un nom de concept primitif soit un nom de concept non-primitif dont la définition est une expression conjonctive de concept.

*Exemple 4:*  $CN \sqcap C \sqsubseteq D \iff CN \sqsubseteq D \sqcup \neg C$

## 2.5 Heuristique

En exécutant des algorithmes du raisonnement, l'heuristique peut être employée à essayer de trouver un "bon" ordre dans lequel pour appliquer des règles d'inférence (nous appellerons ces règle-ordre heuristiques) et, pour les règles non-deterministic, un ordre qui peut explorer les différents choix d'expansion est obtenu par les applications de ces règles (nous appellerons ces expansion-ordre heuristiques). Le but est de choisir un ordre qui mène rapidement à la découverte d'un modèle (au cas où l'entrée serait satisfaite) ou à une preuve qu'aucun modèle n'existe (au cas où l'entrée serait insatisfaite).

On présente succinctement quelques méthodes d'heuristique qui sont réalisées dans le problème SAT:

Cet heuristique est simple, facile pour mettre en application, assez précis, et aussi un problème-indépendant. Son but est de trouver des propositions ayant des occurrences maximum dans les clauses de tailles minimums - par conséquent son nom. Plus formellement, on note par  $f_n(l)$  le nombre des occurrences d'un "open literal  $l$ " dans la clause ouverte de longueur  $n$  et par  $f^*(l)$  le nombre des occurrences de  $l$  dans les clauses ouvertes qui sont les plus courtes. Le but de cet heuristique est de choisir un "literal ouvert  $l$ " tel que:

- i)  $f^*(l)$  est maximum
- ii)  $f^*(\bar{l})$  est maximum
- iii) Le minimum de ces deux quantités devrait être maximal

Une des heuristiques les plus répandues dans le problème SAT est le MOMS (Maximum Occurrences in clauses of Minimum Size) [9]. Cet heuristique est simple, facile pour mettre en application, assez précis, et aussi un problème-indépendant. Son but dans le raisonnement de DL est simplement de trouver les expressions ayant des occurrences maximums d'un disjoint dans les disjonctions de tailles minimums.

## 3 Logique de Description Distribuée

Logique de Description Distribuée (LDD) est proposé par Borgida et Serafini [7] pour représenter et raisonner au sujet des bases de connaissance dans les environnements distribués. Nous rappelons brièvement quelques définitions de DDL comme données dans [7].

### 3.1 Syntaxe

Etant donné  $\{\mathcal{LD}_i\}_{i \in I}$ , une collection des logiques de description, où  $I$  est un ensemble non vide des indexes. Pour chaque  $i \in I$ , un TBox  $\mathcal{T}_i$  est présenté dans une concrète  $\mathcal{LD}_i$ . Afin de distinguer des descriptions dans chaque TBox  $\mathcal{T}_i$ , nous mettons en tête des descriptions avec l'index de leurs TBoxes. Par exemple,  $i : C$  dénote un concept  $C$  de  $\{\mathcal{LD}_i\}_{i \in I}$ . Les morphismes sémantiques entre les différents TBoxes sont présentés en utilisant des règles de pont.

**Definition 1** (règle de pont) Une règle de pont de  $\mathcal{T}_i$  à  $\mathcal{T}_j$  est une expressions d'une des trois formes suivants:

- (1)  $i : x \sqsubseteq j : y$ , règle de pont "into"
- (2)  $i : x \sqsupseteq j : y$ , règle de pont "onto"
- (3)  $i : x \equiv j : x$ , règle de pont identique

Où  $x$  et  $y$  sont soit eux concepts soit deux rôles, et soit deux individus de  $\mathcal{LD}_i$  et  $\mathcal{LD}_j$  respectivement.

Les règles de pont de  $\mathcal{T}_i$  à  $\mathcal{T}_j$  représentent des relations entre  $\mathcal{T}_i$  et  $\mathcal{T}_j$  de point de vue du TBox  $j$ -ième. En particulier, la règle de pont into  $i : A \sqsubseteq j : G$  exprime que, de point de vue de  $\mathcal{T}_j$ , le concept  $A$  dans  $\mathcal{T}_i$  est moins général que son concept local  $G$ .

**Definition 2** (TBox distribué) Un TBox distribué (TBD)  $\mathfrak{T} = (\{\mathcal{T}_i\}_{i \in I}, \mathcal{B})$  se compose d'une collection de TBoxes  $\{\mathcal{T}_i\}_{i \in I}$  et d'une collection des règles de pont  $\mathcal{B} = \{\mathcal{B}_{ij}\}_{i \neq j \in I}$  entre eux.

### 3.2 Sémantique

Chaque TBox  $\mathcal{T}_i$  est interprété par une interprétation locale  $\mathcal{I}_i$  dans son domaine local. Les règles de pont sont interprétées en utilisant la relation de domaine  $r_{ij}$  entre les domaines.

**Definition 3** (Relation de domaine) Une relation de domaine  $r_{ij}$  de  $\Delta^{\mathcal{T}_i}$  à  $\Delta^{\mathcal{T}_j}$  est un sous-ensemble de  $\Delta^{\mathcal{T}_i} * \Delta^{\mathcal{T}_j}$ . On note:  $r_{ij}(d) = \{d' \in \Delta^{\mathcal{T}_j} \mid (d, d') \in r_{ij}\}$ ;  $r_{ij}(D) = \cup_{d \in D} r_{ij}(d)$ , et  $r_{ij}(R) = \cup_{(d, d') \in R} r_{ij}(d) * r_{ij}(d')$ , avec  $D \subseteq \Delta^{\mathcal{T}_i}$  et  $R \subseteq \Delta^{\mathcal{T}_i} * \Delta^{\mathcal{T}_j}$ .

La relation de domaine représente la possibilité d'envoyer des individus de  $\Delta^{\mathcal{T}_i}$  dans  $\Delta^{\mathcal{T}_j}$  du point de vue de  $\mathcal{LD}_j$ .

**Definition 4** (Interprétation distribuée) Une interprétation distribuée  $\mathfrak{J} = (\{\mathcal{I}_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I})$  d'un TBD combine des interprétations locales  $\mathcal{I}_i$  de chaque  $\mathcal{T}_i$  sur le domaine local  $\Delta^{\mathcal{T}_i}$  et une famille des relations  $r_{ij} \subseteq \Delta^{\mathcal{T}_i} * \Delta^{\mathcal{T}_j}$  entre des domaines locaux.

On dit qu'une interprétation  $\mathfrak{J} = (\{\mathcal{I}_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I})$  satisfait les éléments d'un TBD  $\mathfrak{T} = (\{\mathcal{T}_i\}_{i \in I}, \mathcal{B})$  si:

- $\mathfrak{J} \models_d i : A \sqsubseteq j : G$  si  $r_{ij}(A^{\mathcal{T}_i}) \subseteq G^{\mathcal{T}_j}$
- $\mathfrak{J} \models_d i : B \sqsupseteq j : H$  si  $r_{ij}(B^{\mathcal{T}_i}) \supseteq H^{\mathcal{T}_j}$
- $\mathfrak{J} \models_d i : A \sqsubseteq B$  si  $\mathcal{I}_i \models A \sqsubseteq B$
- $\mathfrak{J} \models_d \mathcal{T}_i$  si  $\mathcal{I}_i \models \mathcal{T}_i$
- $\mathfrak{J} \models_d \mathfrak{T}$ , si  $\mathfrak{J} \models_d \mathcal{T}_i$  et  $\mathfrak{J} \models_d \mathcal{B}_{ij}, \forall i, j \in I$
- $\mathfrak{T} \models_d i : A \sqsubseteq B$  si  $\forall \mathfrak{J}, \mathfrak{J} \models_d \mathfrak{T} \implies \mathfrak{J} \models_d i : A \sqsubseteq B$

## 4 Décomposition

Les techniques présentées ne sont vraiment que efficace dans certains cas avec les structures particulières de ICGs. Nos travaux éliminent ICGs de l'ontologie générale (présenté par un TBox) autant que possible en décomposant une ontologie en plusieurs sous-ontologies (présentées par un TBox distribué). Le raisonnement est alors mis en application sur ces sous-ontologies (TBox distribué). L'effet du raisonnement dépend de la méthode de décomposition de TBox original. En conséquence, nous proposons une technique appelée la "décomposition overlay".

### 4.1 Définitions

$(\{\mathcal{T}_i\}, \mathcal{B}_{ij})_{i,j \in I}$  (noté par  $\mathfrak{T}$ ) est une décomposition overlay d'un TBox  $\mathcal{T} = (\mathbf{C}, \mathbf{R}, \mathbf{B}, \mathbf{A})$ , où  $\mathbf{C}, \mathbf{R}, \mathbf{B}, \mathbf{A}$  sont respectivement des ensembles des concepts primitifs, des rôles primitifs, des noms des concepts définis et des axiomes, si:

- Chaque composant  $\mathcal{T}_i$  se compose de tous les concepts primitifs, des rôles primitifs et des concepts définis de  $\mathcal{T}$ , c.à.d.,  $\mathcal{T}_i = (\mathbf{C}, \mathbf{R}, \mathbf{B}, \mathbf{A}_i)$

- $\mathbf{B} = \{B \mid B \doteq C\}$ ,  $\mathbf{A} = \{(C \sqsubseteq D)\}$ ,  $\mathbf{A}_i = \{(C_i \sqsubseteq D_i)\}$ , où  $i \in I$ ;  $B$  est un nom de concept,  $C, D, C_i$  et  $D_i$  sont des expressions de concept.

- $\mathbf{A}_i \subseteq \mathbf{A}$ ;  $\cup_i \mathbf{A}_i = \mathbf{A}$  et  $\cap_i \mathbf{A}_i = \emptyset$

- $\mathcal{B}_{ij}$  est un ensemble des applications sémantiques de  $\mathcal{T}_i$  dans  $\mathcal{T}_j$ . Une application sémantique est une relation identique entre deux concepts (deux rôles) qui apparaissent également dans deux axiomes de TBoxes différents ( $\mathcal{B}_{ij}$  peut être vide si  $\mathbf{A}_i$  et  $\mathbf{A}_j$  sont "disjoints"<sup>1</sup>).

Ainsi, une décomposition overlay d'un TBox  $\mathcal{T}$  est un TBox distribué "spécial". Il est appelé "special" car  $\mathcal{B}_{ij}$  inclut seulement les règles de pont identiques. Ils ont également toutes les propriétés de TBox distribué général (comme donné la section ci-dessus) et quelques propriétés particulières (seront proposé dans une section suivante).

Nous devons donner une stratégie pour décomposer  $\mathcal{T}$  en des sous-ensembles  $\{\mathcal{T}_i\}$  tels que chaque  $\mathcal{T}_i$  est une forme de LD d'une certaine ontologie.

Nous déterminons l'ensemble des applications sémantiques et les représentons sous forme de règles de pont dans LDD. On utilise  $\mathcal{B}$  pour indiquer la totalité des règles de pont obtenues.

*Exemple 5:*  $i : X \equiv j : X$ ,

où,  $X$  est un concept (un rôle),  $i \neq j, i, j \in \{1, 2\}$ ;  $i : X$  dénote un concept (rôle)  $X$  de TBox  $\mathcal{T}_i$

L'essence de cette décomposition est seulement d'examiner les axiomes, c.à.d., l'ensemble des axiomes de TBox original est divisé en plusieurs sous-ensembles des axiomes pour sous-TBoxes respectivement. Nous sommes concernés les axiomes parce qu'ils influencent directement le problème de raisonnement. Cependant, le

---

<sup>1</sup>  $\mathbf{A}_i$  et  $\mathbf{A}_j$  sont appelés disjoints si il n'existe pas d'un concept (un rôle) commun dans tous les deux  $\mathbf{A}_i$  et  $\mathbf{A}_j$ .

grand nombre des concepts et des rôles est également une difficulté pour maintenir, réutiliser et développer indépendamment des ontologies composantes.

*Exemple 6: Décomposer  $\mathcal{T}$  en  $\mathcal{T}_1, \mathcal{T}_2$  et  $\mathcal{T}_3$*

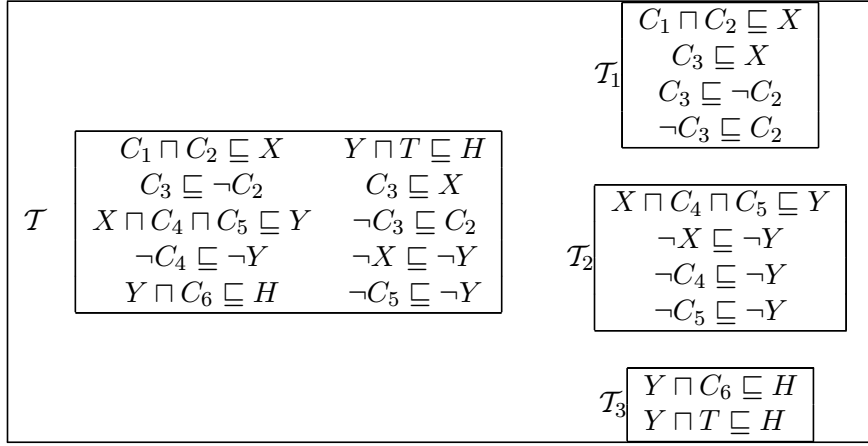


Figure 1: Une décomposition de TBox  $\mathcal{T}$  et leurs trois sous-TBoxes

La droite de la Figure 1 décrit une décomposition de TBox  $\mathcal{T}$  en trois TBoxes  $\mathcal{T}_1, \mathcal{T}_2$  and  $\mathcal{T}_3$

#### 4.2 Propriétés suppléments de TBox distribué dans la décomposition

La décomposition d'un TBox est également un TBox distribué, elle conserve donc toutes les propriétés d'un TBox distribué général. De plus elle a quelques autres:

Nous pouvons ajouter au sujet des interprétations "satisfaisantes" TBoxes:

$$\mathfrak{J} \models_d i : A \sqsubseteq j : A \text{ si } r_{ij}(A^{\mathcal{I}_i}) \equiv A^{\mathcal{I}_j}$$

$$\mathfrak{J} \models_d i : A \sqsubseteq j : B \text{ si } r_{ij}(A^{\mathcal{I}_i}) \subseteq B^{\mathcal{I}_j}$$

$$\mathfrak{I} \models_d i : A \sqsubseteq j : B \text{ si, pour chaque } \mathfrak{J}, \mathfrak{J} \models_d \mathfrak{I} \implies \mathfrak{J} \models_d i : A \sqsubseteq j : B$$

Remarquons que, tous  $\mathcal{I}_i, (i \in I)$  ont le même ensemble des concepts et des rôles de  $\mathcal{T}$ , et la notation  $i : X$  est de prouver que nous parlons de  $X$  dans  $\mathcal{I}_i$ .

#### 4.3 Propriétés de Décomposition

Notre solution est visée à éliminer GCIs d'un TBox autant que possible en décomposant une ontologie en plusieurs sous-ontologies. Le raisonnement est alors réalisé sur ces sous-ontologies.

- Quand une disjonction dans le label de noeud est développée, l'ordre dans lequel des disjonctifs sont choisis peut faire un changement énergétique de la performance du raisonneur de tableau

- On va diviser une ontologie en deux sous-ontologies tels que deux ensembles des axiomes dans deux ces ontologies soient le plus "incohérent" possible.

- Les nombres des axiomes dans ces deux sous-ontologies sont les plus équilibrés possibles.

Avant de réaliser une décomposition, on donne quelques définitions principales dans le suivant.

Intuitivement, une décomposition  $(\{\mathcal{T}_i\}, \mathcal{B}_{ij})$  de  $\mathcal{T}$  est "bonne" si: la quantité des objets communs (des concepts et des rôles) dans les ontologies décomposées est la plus petite que possible et les nombres des axiomes dans deux sous-ontologies sont les plus équilibrés possibles. Les ontologies décomposées sont consistents, elles ont le même domaine que celui de l'ontologie originale, parce qu'elles reproduisent tous les concepts et tous les rôles de l'ontologie originale. Nous impliquons facilement que les ontologies décomposées ont également la même interprétation que celle de l'ontologie originale. En outre, elle doit avoir les propriétés suivantes:

(1) La conservation des concepts (rôles): En effet, chaque concept (rôle) de  $\mathcal{T}$  est également un concept (rôle) dans tous les  $\mathcal{T}_i$ .

(2) La conservation des axiomes:  $\mathbf{A} \subseteq \cup_i \mathbf{A}_i$ , pour chaque axiome  $C \sqsubseteq D$  de l'ontologie  $\mathcal{T}$ , c'est également un axiome dans un des  $\mathcal{T}_i$ . Notons que, il n'y a pas d'axiome qui se produit dans deux ontologies différentes (c.-à-d., les ensembles des axiomes de deux ontologies différentes sont disjoints).

(3) On propose les applications sémantiques afin de conserver les relations des entités entre sous-ontologies, elles présentent seulement des relations identiques entre deux concepts (deux rôles) qui se produisent dans deux axiomes de deux ontologies différentes. Elles sont présentées par les règles de pont  $\mathcal{B}$ .

Ces trois propriétés doivent facilement déduire de la définition de la décomposition, elles prouvent la conservation de la syntaxe. Elles assurent que  $(\{\mathcal{T}_i\}, \mathcal{B})$  est bien représenté par LDD.

(4) La conservation de la sémantique, c.à.d.,  $\Delta = \Delta_i$ , et  $\cdot^{\mathcal{I}} = \cdot^{\mathcal{I}_i}, \forall i \in I$ ;  $\mathcal{I}, \mathcal{I}_i$  sont des interprétations de  $\mathcal{T}$  et  $\mathcal{T}_i$  respectivement.

(5) La conservation des services d'inférence, c.à.d, si une description de concept est satisfaite par rapport à (p.r.à)  $\mathcal{T}$  alors elle est également satisfaite p.r.à  $(\{\mathcal{T}_i\}, \mathcal{B})$  et réciproquement.

Outre ces propriétés, nous proposons deux approches du raisonnement avec des ontologies décomposées. En caractérisant la forme de décomposition, nous prouverons qu'il existe d'un algorithme déterminé pour décomposer une ontologie représentée par LD en plusieurs ontologies représentées par LDD qui conforment à cette forme. Dans cette partie, nous nous concentrons seulement sur deux propriétés qui sont les plus importantes de la décomposition, elles se composent des conservations de la sémantiques et de l'inférence. Les définitions et les preuves suivantes sont seulement examinées dans un TBD  $(\{\mathcal{T}_i\}, \mathcal{B})$  de  $\mathcal{T}$ , c.à.d.,  $\mathcal{T}$  et  $\{\mathcal{T}_i\}$  ont les mêmes concepts et les mêmes rôles. Par conséquent, au lieu de dire "des concepts (rôles) communs", nous disons brièvement "des concepts (rôles)".

**Definition 5** (*Expansion*)

L'expansion d'une description de concept  $C$  peut se comprendre comme l'ensemble des concepts construisant  $C$ . Formellement, l'expansion d'un concept, d'une description de concept, d'un axiome est définie périodiquement comme suit:

Expansion  $Ex(.)$  est une application qui envoie un concept (une description de concept ou un axiome) dans l'ensemble des concepts primitifs et des rôles primitifs qui se produisent dans leurs définitions:

- Si  $A$  est un concept (un rôle) atomique ou sa négation, alors  $Ex(A) = \{A\}$
- Si  $A$  est un concept défini en forme  $\exists R.C$  ou  $\forall R.C$ , alors  $Ex(A) = \{A\} \cup Ex(R) \cup Ex(C)$
- Si  $A$  est un concept défini en forme  $C_1 \sqcap C_2$  ou  $C_1 \sqcup C_2$ , alors  $Ex(A) = \{A\} \cup Ex(C_1) \cup Ex(C_2)$
- Si  $A$  est une description composite de concept en forme  $\rho(M_1, \dots, M_k)$ , où  $\rho$  est un constructeur de concept,  $Ex(\rho(M_1, \dots, M_k)) = \cup_i \{Ex(M_i)\}$ .
- Pour un axiome  $C \sqsubseteq D$ ,  $Ex(C \sqsubseteq D) = Ex(C) \cup Ex(D)$ .

**4.3.1 Incohérence**

1.1 *Au point de vue de syntaxe:*

- Deux concepts  $C$  et  $D$  sont appelés incohérent si:
  - +  $Ex(C) \cap Ex(D) = \phi$  ou
  - +  $Ex(C)$  contient un concept  $E$  et  $Ex(D)$  contient  $\neg E$  ou
  - +  $\neg C$  apparaît dans définition de  $D$  (ou inversement)

*Exemple 7:*  $D \equiv \neg C \sqcap C_i$ , où  $C_i$  est une expression de concept, rôle,...

- Deux axiomes  $C_i \sqsubseteq D_i$  et  $C_j \sqsubseteq D_j$  sont appelés incohérents si  $Ex(C_i \sqsubseteq D_i) \cap Ex(C_j \sqsubseteq D_j) = \phi$  ou  $C_i \sqsubseteq D_i$  contient une expression de concept  $E$  et  $C_j \sqsubseteq D_j$  contient une expression de concept de négation  $\neg E$  (ou inversement)
- Deux ontologies  $\mathcal{O}_i$  et  $\mathcal{O}_j$  sont appelés incohérents si deux ensembles des axiomes dans  $\mathcal{O}_i$  et  $\mathcal{O}_j$  sont incohérents  $Ex(\{C_i \sqsubseteq D_i\}) \cap Ex(\{C_j \sqsubseteq D_j\}) = \phi$

1.2 *Au point de vue de sémantique*

- Deux concepts  $C$  et  $D$  sont appelés incohérents s'il n'existe pas d'individu  $x$  tel que  $x \in C^{\mathcal{I}}$  et  $x \in D^{\mathcal{I}}$ ,  $\forall$  modèle  $\mathcal{I}$  de  $\mathcal{O}$
- Deux axiomes  $C_i \sqsubseteq D_i$  et  $C_j \sqsubseteq D_j$  sont appelés incohérents s'il n'existe pas d'individu  $x$  tel que  $x$  satisfait tous les deux  $C_i \sqsubseteq D_i$  et  $C_j \sqsubseteq D_j$  (c.a.d., il n'existe pas  $x$  tel que  $x \in (\neg C_i \sqcup D_i)^{\mathcal{I}}$  et  $x \in (\neg C_j \sqcup D_j)^{\mathcal{I}}$ ,  $\forall$  modèle  $\mathcal{I}$  de  $\mathcal{O}$ ).
- Deux ensembles des axiomes  $\{C_i \sqsubseteq D_i\}$  et  $\{C_j \sqsubseteq D_j\}$  sont appelés incohérents si deux axiomes  $C_i \sqsubseteq D_i$  et  $C_j \sqsubseteq D_j$  sont deux à deux incohérents.

Les définitions au-dessus sont strictement incohérentes. En réalité, il n'y a pas beaucoup de paire de tels concepts (axiomes, les ensembles des axiomes). Une décomposition d'une ontologie en plusieurs ontologies telles que ces ontologies soient

strictement incohérentes est très difficile et on ne peut pas presque réaliser. Donc, on doit donner une mesure de degré d'incohérence.

*1.3 Mesure de degré d'incohérence des axiomes (selon l'ordre dégressif)*

- Deux axiomes complets contiennent des paires des concepts incohérents.
- Deux axiomes contiennent le même conjonct.

#### 4.3.2 Cohérence

- Le concept cohérent est un concept qui apparaît plusieurs fois dans des axiomes.
  - Deux concepts  $C$  et  $D$  sont cohérents si  $C$  apparaît dans la définition de  $D$  (ou inversement)

*Exemple 8:*  $D \equiv C \sqcap C_i \sqcup D_i$ , où  $C_i, D_i$  sont des expressions de concept, rôle,...

(Deux concepts  $C$  et  $D$  sont strictement cohérents si  $C$  est un conjonct dans la définition de  $D$ , et  $D$  est une conjonction des concepts, c.à.d. il y a seulement un operateur de conjonction dans la définition de  $D$  (ou inversement).

*Exemple 9:*  $D \equiv C \sqcap C_1 \sqcap \dots \sqcap C_n$  )

- Deux axiomes  $A_1$  et  $A_2$  sont cohérents s'ils ont la même expression de concept ou ont deux concepts cohérents dans la même côté (droite ou gauche).

*Exemple 10:*  $A_1 : C \sqcap C_i \sqsubseteq D_i$

$A_2 : C \sqcap C_j \sqsubseteq D_j$  ou

$A_1 : C_i \sqsubseteq D_i \sqcap E_i$

$A_2 : C_j \sqsubseteq D_j \sqcap E_j$

où  $E_i$  et  $E_j$  sont deux concepts cohérents

Le degré de la cohérence dégressive selon l'ordre suivant:

- + ces axiomes contiennent plusieurs conjonctions ressemblantes.
- + ces axiomes contiennent des concepts  $\exists.R$  qui ont le même  $R$
- + ces axiomes contiennent des concepts et des rôles ressemblants.

**Definition 6** On définit une fonction de décomposition  $\#()$  (comme donnée dans [7]) des concepts et des rôles de Tbox original  $\mathcal{T}$  en des concepts et des rôles dans  $\mathcal{T}_i$  ( $i \in I$ ) comme suivants :

1.  $\#(M) = (i, M)$  pour tous les concepts primitifs et tous les rôles primitifs.

2. Si  $\rho$  est un constructeur de concept (un constructeur de rôle) avec  $k$  arguments, alors

$$\#(\rho(M_1, \dots, M_k)) = \rho(\#(i, M_1), \dots, \#(i, M_k)).$$

*Exemple 11:*  $\#(C_1 \sqcap \forall R.C_2)$  produit  $i : C_1 \sqcap \forall (i : R).(i : C_2)$

Nous fournissons maintenant un TBD en LDD. Premièrement, ANYTHING, NOTHING dénotent les concepts "top" et "bottom" de  $\mathcal{T}$  respectivement, ils sont distingués à partir des concepts  $\text{Top}_i$  et  $\text{Bot}_i$  de  $\mathcal{T}_i$ .

**Definition 7** L'application  $\#$  pour un TB global  $\mathcal{T}$ , produit un TBD  $\#(\mathcal{T}) = (\{\mathcal{T}_i\}_{i \in I}, \mathcal{B}_{ij})$  dans la langage LDD, où  $\mathcal{B}_{ij}$  est l'ensemble des règles de pont identiques, se compose des axiomes suivants :

1. (\*on copie tous les axiomes de TBox global au même TBox local  $\mathcal{T}_i, i \in I^*$ )  
 $i : \#(X) \sqsubseteq \#(Y)$  , pour tous  $X \sqsubseteq Y \in \mathcal{T}$  et  $\#(X) \in \mathcal{T}_i ; \#(Y) \in \mathcal{T}_i$
2.  $i : \#(X) \equiv j \#(X)$ , pour tous  $X \in \mathcal{T}$  ,  $\#(X) \in \mathcal{T}_i ; \#(X) \in \mathcal{T}_j$  et  $X \in Ex(\mathbf{A}_i), X \in Ex(\mathbf{A}_j)$
3.  $ANYTHING \sqsubseteq Top_i$  (\*assurant que  $Top_i$  n'est pas vide\*)
4.  $Bot_i \sqsubseteq NOTHING$  (\*limitant  $Bot_i$  être un concept incohérent\*)
5.  $(i, A) \sqsubseteq Top_i$  (\*assurant que  $Top_i$  est le Top local approprié de son hiérarchie IS-A\*)
6. (\*assurant que chaque rôle  $R$  est dans le même domaine et le même espace de  $Top_i$  \*)  
 $Top_i \sqsubseteq \forall(i, R).(Top_i)$  pour chaque rôle  $R$  de  $\mathcal{T}$  (\* l'espace de  $(i, R)$  est dans  $\Delta^{\mathcal{I}_i}$  \*)  
 $\exists(i, R).ANYTHING \sqsubseteq Top_i$  (\*  $R$  est seulement défini dans  $\Delta^{\mathcal{I}_i}$  \*)

**Definition 8** Etant donné deux Tboxes  $\mathcal{T}_i$  et  $\mathcal{T}_j, i \neq j$ , avec leurs interprétations  $\mathcal{I}_i = (\Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i})$  et  $\mathcal{I}_j = (\Delta^{\mathcal{I}_j}, \cdot^{\mathcal{I}_j})$  respectivement. Nous disons que  $\mathcal{T}_i$  et  $\mathcal{T}_j$  sont interprétés dans le même domaine (ou  $\mathcal{I}_i$  et  $\mathcal{I}_j$  ont le même domaine) ssi  $\Delta^{\mathcal{I}_i} = \Delta^{\mathcal{I}_j}$  et  $\cdot^{\mathcal{I}_i} = \cdot^{\mathcal{I}_j}$  pour tous leurs concepts (rôles) primitif communs.

**Definition 9** Etant donné  $\rho$  un constructeur de concept dont la sémantique peut être exprimée comme  $\rho(\arg_1, \dots, \arg_n)^{\mathcal{I}} = f_\rho(\arg_1^{\mathcal{I}}, \dots, \arg_n^{\mathcal{I}}, \Delta^{\mathcal{I}})$ , où  $f_\rho(X_1, \dots, X_n, DY)$  est une fonction dont la définition ne contient aucune référence à  $\mathcal{I}$ . Soit  $B_1, \dots, B_n, W, \Delta$  des ensembles tels que  $W \subseteq \Delta$ , et chaque  $B_j$  est un sous-ensemble de  $W$  ou de  $WxW$ ,  $1 \leq j \leq n$ . Alors  $\rho$  est appelé un constructeur local si  $f_\rho$  satisfait :  $f_\rho(B_1, \dots, B_n, \Delta) = f_\rho(B_1, \dots, B_n, W)$

quand il est un constructeur de concept ou de rôle.

**Corollaire:** Si deux Tboxes  $\mathcal{T}_i$  et  $\mathcal{T}_j$  sont interprétés dans le même domaine, alors nous avons  $C^{\mathcal{I}_i} = C^{\mathcal{I}_j}$ , pour chaque  $C$  est une description de concept commun arbitraire de  $\mathcal{T}_i$  et  $\mathcal{T}_j$ .

**Proof.** - Parce que  $\mathcal{T}_i$  et  $\mathcal{T}_j$  sont interprétés dans le même domaine, nous avons  $C^{\mathcal{I}_j} \equiv C^{\mathcal{I}_i}$ ,  $R^{\mathcal{I}_j} \equiv R^{\mathcal{I}_i}$  pour  $C, R$  sont un concept primitif commun and un rôle primitif commun respectivement. En induisant sur la structure d'un concept arbitraire (rôle)  $B$ , nous montrons facilement que  $B^{\mathcal{I}_j} \equiv B^{\mathcal{I}_i}$ .

- Pour une description de concept commun arbitraire  $M$ , ils seront construit des concepts (rôles) primitifs communs. Supposez  $M$  ayant la forme  $\rho(M_1, \dots, M_k)$ , où des concepts  $M_i, \dots, M_k$  sont moins complexes que  $M$ , et ils apparaissent également dans  $\mathcal{T}_j$ ,  $\rho$  est un constructeur de concept avec  $k$  arguments.

Par induction structurale sur  $M$ , nous obtenons  $M_p^{\mathcal{I}_j} \equiv M_p^{\mathcal{I}_i}$  pour tous  $p = 1, \dots, k$ . Nous devons prouver que  $M^{\mathcal{I}_i} = M^{\mathcal{I}_j}$ .

En effet,

$$M^{\mathcal{I}_j} \equiv (\rho(M_1, \dots, M_k))^{\mathcal{I}_j} \equiv f_\rho(M_1^{\mathcal{I}_j}, \dots, M_k^{\mathcal{I}_j}, \Delta^{\mathcal{I}_j}) \equiv f_\rho(M_1^{\mathcal{I}_i}, \dots, M_k^{\mathcal{I}_i}, \Delta^{\mathcal{I}_i}) \equiv (\rho(M_1, \dots, M_k))^{\mathcal{I}_i} \equiv M^{\mathcal{I}_i} . \blacksquare$$

## 5 Raisonnement Parallèle

Nous examinons le plus simple cas, une décomposition de TBox  $\mathcal{T}$  en deux TBoxes  $\mathcal{T}_1$  et  $\mathcal{T}_2$ .

Pour vérifier la satisfaction d'un concept  $C$  par rapport à (p.r.à)  $\mathcal{T}$  et  $\{\mathcal{T}_1, \mathcal{T}_2\}$ , on réalise un algorithme de tableaux normal sur TBox  $\mathcal{T}_1$ , on appelle  $T_1(C)$ , et on réalise en même temps un algorithme de tableaux sur TBox  $\mathcal{T}_2$ , on appelle également  $T_2(C)$ . Nous allons obtenir des résultats suivants:

1. Si  $T_1(C)$  ou  $T_2(C)$  est insatisfait alors on conclut que  $T(C)$  est insatisfait p.r.à  $\mathcal{T}$ .
2. SiNON, c.à.d,  $T_1(C)$  et  $T_2(C)$  sont tous les deux satisfaits alors on va faire un fusionnement (merging) des  $T_1$  et  $T_2$ .
  - Fusionnement des  $T_1$  et  $T_2$

Après réaliser le calcul de tableau sur  $T_1$  et  $T_2$ , on trouve deux arbres de modèle  $AM_1$  et  $AM_2$  respectivement. On va fusionner des noeuds de feuille dans  $AM_1$  et  $AM_2$  comme suivant: d'abord on passe sur des noeuds de  $AM_1$  et de  $AM_2$  qui sont contradictoires, on va fusionner deux à deux chaque noeud dans  $AM_1$  avec des noeuds dans  $AM_2$ . On suppose que  $AM_1$  ait  $m$  noeuds de feuille,  $AM_2$  ait  $n$  noeuds de feuille (où  $m, n$  sont des nombres entiers non-négatifs), alors on obtient les noeuds nouveaux  $m.n$ . Si tous ces noeuds  $m.n$  sont contradictoires alors on conclut que  $C$  est satisfait p.r.à  $\{\mathcal{T}_1, \mathcal{T}_2\}$ , sinon on conclut que  $C$  est insatisfait p.r.à  $\{\mathcal{T}_2, \mathcal{T}_2\}$  (c.a.d., s'il existe au moins un noeud qui est non-contradictoire, alors on conclut que  $C$  est insatisfait).

Fusionner deux noeuds  $x_{T_1}$ -label  $\mathcal{L}(x_{T_1}^C)$  et  $x_{T_2}$ -label  $\mathcal{L}(x_{T_2}^C)$  en un noeud  $x_{T_{12}}$ -label  $\mathcal{L}(x_{T_{12}}^C) = \mathcal{L}(x_{T_1}^C) \cup \mathcal{L}(x_{T_2}^C)$ , selon les principes suivants:

- i)  $\exists R.C_1 \in \mathcal{L}(x_{T_1}^C)$  et  $\exists R.C_1 \in \mathcal{L}(x_{T_2}^C)$ , alors on va créer un noeud  $x_{T_{12}}$ -label  $\mathcal{L}(x_{T_{12}}^C) = \mathcal{L}(x_{T_1}^C) \cup \mathcal{L}(x_{T_2}^C)$ , et un bord -label  $R$  entre  $x_{T_{12}}$  et un noeud  $y : C_1$
- ii)  $\exists R.C_1 \in \mathcal{L}(x_{T_1}^C)$  et  $\exists R.C_2 \in \mathcal{L}(x_{T_2}^C)$ , alors on va créer un noeud  $x_{T_{12}}$ -label  $\mathcal{L}(x_{T_{12}}^C) = \mathcal{L}(x_{T_1}^C) \cup \mathcal{L}(x_{T_2}^C)$ , et un bord -label  $R$  entre  $x_{T_{12}}$  et un noeud  $y : C_1, C_2$
- iii)  $\exists R_1.C_1 \in \mathcal{L}(x_{T_1}^C)$  et  $\exists R_2.C_2 \in \mathcal{L}(x_{T_2}^C)$ , alors on va créer un noeud  $x_{T_{12}}$ -label  $\mathcal{L}(x_{T_{12}}^C) = \mathcal{L}(x_{T_1}^C) \cup \mathcal{L}(x_{T_2}^C)$ , et deux bords que l'un-label  $R_1$  entre  $x_{T_{12}}$  et un noeud  $y_1 : C_1$ , et l'autre-label  $R_2$  entre  $x_{T_{12}}$  et un noeud  $y_2 : C_2$
- iv)  $\exists R.C_1 \in \mathcal{L}(x_{T_1}^C)$ ,  $\forall S.C_2 \in \mathcal{L}(x_{T_2}^C)$  et  $R \sqsubseteq S$ , alors on va créer un noeud  $x_{T_{12}}$ -label  $\mathcal{L}(x_{T_{12}}^C) = \mathcal{L}(x_{T_1}^C) \cup \mathcal{L}(x_{T_2}^C)$ , et un bord -label  $R$  entre  $x_{T_{12}}$  et un noeud  $y : C_1, C_2$ .

où  $R_1, R_2$  et  $S$  sont des noms de rôle;  $C_1$  et  $C_2$  sont des noms de concept.

Example 12: Merging two nodes with labels  $\{C, D, \exists R.C_2, \forall R.C_3\}$  and  $\{C, \exists R.C_4, \forall R.C_3, \exists R_1.C_5\}$  of trees  $\mathbf{T}_1$  and  $\mathbf{T}_2$  respectively (Figure 3)

```

Input:  $Tab_1(C), Tab_2(C)$ 
Output: Satisfiable / Unsatisfiable
1: BEGIN
2:  $\mathbf{T}_1 = Tab_1(C)$ ;
3:  $\mathbf{T}_2 = Tab_2(C)$ ;
4:  $\mathbf{T} = \emptyset; k = 0$ ;
5: For each open branch  $\beta_1$  in  $\mathbf{T}_1$  do
6:   for each open branch  $\beta_2$  in  $\mathbf{T}_2$  do
7:      $\alpha_k = \beta_1 \cup \beta_2$ ;
8:      $\mathbf{T} = \mathbf{T} \cup \alpha_k$ ;
9:      $k = k + 1$ ;
10:  end for
11: end for
12: If (( $\mathbf{T}_1$  is clashed) or ( $\mathbf{T}_2$  is clashed) or ( $\mathbf{T}$  is clashed)) then
13:  return unsatisfiable
14: Else
15:  return satisfiable
16: END.

```

Figure 2: L'algorithme de tableau parallèle

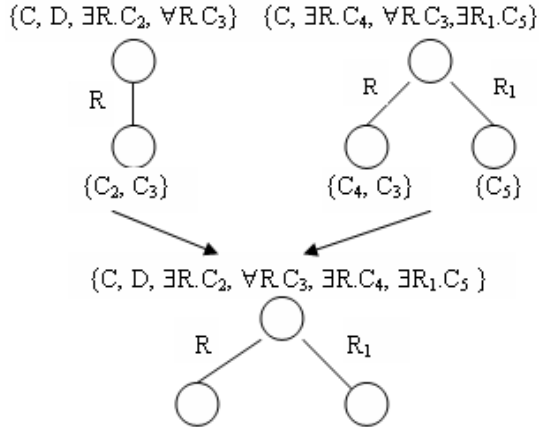


Figure 3: Merging 2 nodes of  $\mathbf{T}_1$  and  $\mathbf{T}_2$

## 6 Raisonnement dans La Logique de Description Distribuée

### 6.1 L'algorithme

Nous examinons le plus simple cas, une décomposition de TBox  $\mathcal{T}$  en deux TBoxes  $\mathcal{T}_1$  et  $\mathcal{T}_2$ .

Pour vérifier la satisfaction d'un concept  $C$  par rapport à (p.r.à)  $\mathcal{T}$  et p.r.à  $(\{\mathcal{T}_1, \mathcal{T}_2\}, \mathcal{B})$ . D'abord nous réalise comme précédente, c.à.d, on réalise l'algorithme de tableau  $\mathbf{T}_1(C)$  et  $\mathbf{T}_2(C)$  sur tous les deux ontologies  $\mathcal{T}_1$  et  $\mathcal{T}_2$  en même temps (en parallèle). On va également trouver deux cas suivants:

1. Si  $\mathbf{T}_1(C)$  ou  $\mathbf{T}_2(C)$  est insatisfait (tous les feuilles de  $\mathbf{T}_1(C)$  ou de  $\mathbf{T}_2(C)$  sont des clashes) alors on conclut que  $C$  est insatisfait p.r.à  $\mathcal{T}$ .

2. SiNON, c.à.d,  $\mathbf{T}_1(C)$  et  $\mathbf{T}_2(C)$  sont ainsi satisfaits (il existe au moins une feuille de  $\mathbf{T}_1(C)$  et de  $\mathbf{T}_2(C)$  sont non-clash) alors on va continuer faire l'algorithme de tableau sur  $\mathcal{T}_2$  pour des noeuds de  $\mathbf{T}_1(C)$  en utilisant des règles de pont, c.à.d,  $\mathbf{T}_2(\mathbf{T}_1(C))$  est appliqué seulement sur des feuilles non-clashes de  $\mathbf{T}_1(C)$ . Nous ajoutons des axiomes de  $\mathcal{T}_2$  au system de contrainte  $\mathbf{T}_2(\mathbf{T}_1(C))$  selon quelques règles de pont entre  $\mathcal{T}_1$  et  $\mathcal{T}_2$ .

```

Input :  $dTab(C)$ 
Output : Satisfiable /Unsatisfiable
1 : BEGIN
2 :  $\mathbf{T} = dTab(C)$  ;
3 : If  $\mathbf{T} = Tab_j(C)$  then {execute the local reasoning
   in  $\mathcal{T}_j$  and generate the complete tree}
4 :   for each open branch  $\beta$  in  $\mathbf{T}$  do
5 :     repeat
6 :       select node  $x \in \beta$  and an  $i \neq j$  ;
7 :        $\phi i dtq(x) = \{D|j : D \rightarrow i : D, D \in L(x)\}$ 
8 :       if ( $\phi i dtq(x) \neq \emptyset$ ) then
9 :         for each  $D \in \phi i dtq(x)$  do
10 :          if ( $dTab_i(D)$ ) is not satisfiable then
11 :            close {clash in  $x$ }
12 :            break; {verify next branch}
13 :          end if
14 :        end for
15 :      end if
16 :    until (( is open) and (there exist not verified nodes in ))
17 :  end for {all branches are verified}
18 : end if
19 : if ( $\mathbf{T}$  is clashed) then
20 :   return unsatisfiable;
21 : else
22 :   return satisfiable;
23 : end if
24 : END.

```

Figure 4: L'algorithme de tableau distribué

**Theorem** (*Correction et Complétion*)

Etant donné  $\mathcal{T}$  un TBox et  $\mathcal{T}_{12} = (\{\mathcal{T}_i\}_{i \in \{1,2\}}, \mathcal{B}_{ij})$  un TBox distribué présentant sa décomposition, alors  $\mathcal{T}_{12} \models_d i : X \sqsubseteq j : Y \Leftrightarrow \mathcal{T} \models X \sqsubseteq Y$ , où  $X, Y$  sont des concepts dans  $\mathcal{T}$ .

**Proof.** ( $\Rightarrow$ ):  $\mathcal{T}_{12} \models_d i : X \sqsubseteq j : Y \Rightarrow \mathcal{T} \models X \sqsubseteq Y$

Soit  $\mathcal{I}$  une interprétation qui satisfait  $\mathcal{T}$  ( $\mathcal{I} \models \mathcal{T}$ ), avec le domaine  $\Delta^{\mathcal{I}}$ , nous devons prouver que  $\mathcal{I} \models X \sqsubseteq Y$  (ou  $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$ ).

En effet,

1. A partir de  $\mathcal{I}$ , on définit  $\mathfrak{J} = (\{\mathcal{I}_i\}, \{r_{ij}\})$ , où  $\mathcal{I}_i$  est une interprétation avec le domaine  $\Delta^{\mathcal{I}_i} = \Delta^{\mathcal{I}}$  (c.à.d.,  $\mathcal{I}, \mathcal{I}_i$  et  $\mathfrak{J}$  sont dans le même domaine) respectivement (pour tous  $i \in I$ ).  $\mathcal{I}_i$  interprète chaque concept (rôle) primitif (ou concept spécial

ANYTHING)  $(i, C)$  de  $\mathcal{T}_i$  selon la règle  $\#(C)^{\mathcal{I}_i} = C^{\mathcal{I}}$  et  $\{r_{ij}\}$  sont des relations identiques, c.à.d.,  $r_{ij}(C^{\mathcal{I}_i}) \equiv C^{\mathcal{I}_j}$ , pour tous règles de pont identiques entre  $\mathcal{T}_i$  et  $\mathcal{T}_j$ .

- Premièrement, il nous faut démontrer que  $\mathcal{I}_i$  et  $\mathfrak{J}$  sont des interprétations de  $\mathcal{T}_i$  et  $\mathcal{T}_{12}$  respectivement.

En effet, nous trouvons facilement que  $(\Delta_i, \cdot^{\mathcal{I}_i})$  est une interprétation de  $\mathcal{T}_i$ , parce que :

$$+ \Delta^{\mathcal{I}_i} \text{ n'est pas vide} \quad (1)$$

$$+ \text{ANYTHING } \mathcal{I}_i = \Delta^{\mathcal{I}_i} \quad (2)$$

$$+ \text{NOTHING}^{\mathcal{I}_i} = \emptyset \quad (3)$$

$$+ M^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i} \text{ pour tous les concepts } M \text{ de } \mathcal{T}_i \quad (4)$$

$$+ R^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i} x \Delta^{\mathcal{I}_i} \text{ pour tous les rôles } R \text{ de } \mathcal{T}_i \quad (5)$$

+ D'après la décomposition overlay, pour chaque concept (rôle) défini  $M$  de  $\mathcal{T}$ , il existe un concept (rôle) défini  $\#M$  dans  $\mathcal{T}_i (i \in I)$  et  $\#(M)^{\mathcal{I}_i} = M^{\mathcal{I}}$  (car  $\mathcal{I}_i$  et  $\mathcal{I}$  sont dans le même domaine).

- En outre, on a  $\#(M)^{\mathcal{I}_i} = M^{\mathcal{I}}$ , pour tout concept (rôle) arbitraire  $M$  dans  $\mathcal{T}_i$  (corollaire ci-dessus) (\*)

$$\text{En conséquence, } \#(M)^{\mathcal{I}_i} \subseteq \#(N)^{\mathcal{I}_j} \Leftrightarrow M^{\mathcal{I}} \subseteq N^{\mathcal{I}} \quad (6)$$

- Pour chaque axiome  $(i, V) \sqsubseteq (i, W)$  de  $\mathcal{T}_i$ , par la décomposition overlay, il existe un axiome  $V \sqsubseteq W$  de  $\mathcal{T}$  équivalent

Comme  $\mathcal{I}$  satisfait  $\mathcal{T}$  et  $(V \sqsubseteq W) \in \mathcal{T}$ , par conséquent  $\mathcal{I} \models V \sqsubseteq W \Rightarrow V^{\mathcal{I}} \subseteq W^{\mathcal{I}}$

D'autre part,  $\#(M)^{\mathcal{I}_i} = \#(N)^{\mathcal{I}_i} \Leftrightarrow M^{\mathcal{I}} = N^{\mathcal{I}}$ , donc  $V^{\mathcal{I}} \subseteq W^{\mathcal{I}} \Rightarrow \#(V)^{\mathcal{I}_i} \subseteq \#(W)^{\mathcal{I}_i} \Leftrightarrow \mathcal{I}_i \models (i, V) \sqsubseteq (i, W)$  (7)

$$\Rightarrow \mathfrak{J} \models_d i : V \sqsubseteq W \quad (8)$$

$$\text{de (1), (2), (3), (4), (5) et (7), on obtient } \mathcal{I}_i \models \mathcal{T}_i \Rightarrow \mathfrak{J} \models_d \mathcal{T}_i \quad (9)$$

- Pour chaque règle de pont identique  $i : V \equiv j : V$

$$\text{Parce que } V^{\mathcal{I}} \equiv V^{\mathcal{I}} \Leftrightarrow \#(V)^{\mathcal{I}_i} \equiv \#(V)^{\mathcal{I}_j} \Rightarrow \mathfrak{J} \models i : V \equiv j : V \quad (10)$$

De (9) et (10), on a  $\mathfrak{J} \models \mathcal{T}_{12}$

2. D'ailleurs, par l'hypothèse et de  $M^{\mathcal{I}} = \#(M)^{\mathcal{I}_i}$  nous déduisons que  $\mathfrak{J} \models i : X \sqsubseteq j : Y \Rightarrow \#(X)^{\mathcal{I}_i} \subseteq \#(Y)^{\mathcal{I}_j}$

$$\Rightarrow X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$$

$$\Rightarrow \mathcal{I} \models X \sqsubseteq Y$$

$$(\Leftarrow) \mathcal{I} \models X \sqsubseteq Y \Rightarrow \mathcal{T}_{12} \models i : X \sqsubseteq j : Y$$

Soit  $\mathfrak{J} = (\{\mathcal{I}_i\}, \{r_{ij}\})$  une d-interprétation qui satisfait  $\mathcal{T}_{12}$ , où  $\mathcal{I}_i$  est une interprétation de  $\mathcal{T}_i$ ,  $\{\mathcal{I}_i\}_{i \in I}$  sont dans le même domaine,  $\{r_{ij}\}_{i \neq j}$  sont des relations identiques entre deux domaines  $\Delta^{\mathcal{I}_i}$  et  $\Delta^{\mathcal{I}_j}$ . Il faut montrer que  $\mathfrak{J} \models i : X \sqsubseteq j : Y \Rightarrow \#(X)^{\mathcal{I}_i} \subseteq \#(Y)^{\mathcal{I}_j}$

On définit une interprétation  $\mathcal{I}$ , avec le domaine  $\Delta \equiv \Delta^{\mathcal{I}_i}, \forall i \in I$ , qui interprète un concept (rôle) primitif  $C$  de  $\mathcal{T}$  selon la règle  $C^{\mathcal{I}} = \#(C)^{\mathcal{I}_i}$ .

Nous devons montrer que  $\mathcal{I} \models \mathcal{T}$

En effet:

- Pour chaque concept (rôle) arbitraire  $M$  de  $\mathcal{T}$ , il existe un concept (rôle) correspondant  $\#M$  dans  $\mathcal{T}_i$  et on a  $M^{\mathcal{I}} = \#(M)^{\mathcal{I}_i}$  (selon la preuve ci-dessus) (11).

- Pour chaque axiome  $V \sqsubseteq W$  de  $\mathcal{T}$ , par la décomposition overlay, il existe un axiome équivalent  $\#V \sqsubseteq \#W$  dans  $\mathcal{T}_i$

Parce que  $\mathfrak{J} \models_d T_{12} \implies \mathfrak{J} \models_d T_i, \forall i \in I \implies \mathcal{I}_i \models T_i$

A partir de  $\#V \sqsubseteq \#W \in \mathcal{T}_i$ , on a  $\mathcal{I}_i \models \#V \sqsubseteq \#W \implies \#(V)^{\mathcal{I}_i} \subseteq \#(W)^{\mathcal{I}_i} \implies V^{\mathcal{I}} \subseteq W^{\mathcal{I}} \implies \mathcal{I} \models V \sqsubseteq W$  (12)

De (11) et (12)  $\implies \mathcal{I} \models \mathcal{T}$

En outre, de  $\mathcal{T} \models X \sqsubseteq Y$  (par l'hypothèse)  $\implies \mathcal{I} \models X \sqsubseteq Y \implies X^{\mathcal{I}} \subseteq Y^{\mathcal{I}} \implies \#(X)^{\mathcal{I}_i} \subseteq \#(Y)^{\mathcal{I}_j} \implies \mathfrak{J} \models i : X \sqsubseteq j : Y$  ■

## 6.2 Méthode

- Chercher deux axiomes  $A_1$  et  $A_2$  qui sont les plus incohérents et sont les plus longs dans  $\mathcal{O}$  et les mettre dans deux sous-ontologies particulières  $\mathcal{O}_1$  et  $\mathcal{O}_2$  respectivement.

- Mettre des axiomes relatifs à  $A_1$  dans  $\mathcal{O}_1$  et des axiomes relatifs à  $A_2$  dans  $\mathcal{O}_2$

- Si on trouve des relations de subsumption entre deux définitions de concept, alors on peut détecter des contradictions plus tôt. Pour le voir, supposons qu'on a déjà su qu'un concept défini  $C$  subsume un concept défini  $D$ . Si pendant la génération de modèle, un élément est contraint par l'instance de tous les deux  $\neg C$  et  $D$ , une contradiction peut être détectée sans étendre des définitions de  $C$  et  $D$ . De plus, cet approche n'est que réalisée si la définition de concept n'est pas étendu avant commencer vérifier la satisfaisabilité.

## 7 Discussion et travaux futurs

Les optimisations précédentes ne sont pas uniformément efficaces. Elles sont seulement efficaces dans certaines bases de connaissance. La décomposition d'une ontologie en plus petites ontologies comme nous avons décrite est très efficace. Elle a amélioré la vitesse des raisonneurs avec toutes les bases de connaissance dans LDs, comme montré par les résultats que nous avons donnés ci-dessus. La performance du raisonnement est influencé par l'algorithme de décomposition et par le genre d'ontologie originale. Nous allons étudier pour trouver un algorithme efficace et optimal de cette décomposition. Ceci dépend encore d'avoir une méthode efficace (et bon marché) pour analyser les caractéristiques probables d'une ontologie donnée. Nous avons également exécuté plus d'expériences avec des BCs très grandes (comme UMLS...) pour montrer l'effet des ontologies décomposées en effectuant le raisonnement. Nous allons projeter exécuter le raisonnement dans les ontologies décomposées en combinant la méthode parallèle et la méthode distribuée. Nous nous embarquerons dans la linéarisation de l'algorithme de décomposition et traiterons efficacement les requêtes en LDD. Aussi bien qu'une solution efficace pour décomposer une ontologie en plusieurs sous-ontologies et raisonner sur ces sous-ontologies

sera proposée.

### Références

- [1] Evren Sirin et al., Optimizing Description Logic Reasoning with Nominals: First Results. *UMIACS Technical Report 2005-64*.
- [2] I. Horrocks. Optimizing Tableau Decision Procedures for Description Logics. *PhD thesis, University of Manchester, 1995*.
- [3] Volker Haarslev and Ralf Möller. High Performance Reasoning with Very Large Knowledge Bases. *DL-2000*
- [4] Volker Haarslev and Ralf Möller. Optimizing TBox and ABox Reasoning with Pseudo Models. *DL-2000*
- [5] Franz Baader et al. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *In Proceedings of the 3rd International Conference, p. 270-281, Cambridge, MA, October 1992*.
- [6] Bernhard Nebel. Terminological Reasoning is Inherently Intractable. *Artificial Intelligence, 43:235-249*
- [7] Borgida A., Serafini L., Distributed Description Logics: Assimilating information from peer sources. *Journal of Data Semantics, 1:153-184, 2003*.
- [8] Serafini L. and Tamin A., DRAGO: Distributed Reasoning Architecture for the Semantic Web. *Technical Report T04-12-05, ITC-irst, December 2004*.
- [9] Jon William Freeman, Improvements to Propositional Satisfiability Search Algorithms. *Dissertation in Computer and Information Science, 1995*.
- [10] Pham Tuan Minh, De la logique de description à la logique de description distribuée: Etude préliminaire de la décomposition de l'ontologie. *Report of Master, Nice-Sophia Antipolis University, September 2005*.
- [11] Eyal Amir, Sheila McIlraith, Partition-Based Logical Reasoning . *In Proceedings Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000), Morgan Kaufmann, San Mateo, CA, 2000, pp. 389-400*.
- [12] Franz Baader and Ulrike Sattler, An Overview of Tableau Algorithms for Description Logics. *In proceedings of Tableaux 2000*.
- [13] Calvanese Diego, Reasoning with Inclusion Axioms in Description Logics: Algorithms and Complexity. *ECAI-96, pp 303-307, 1996*.