Decomposition-based Reasoning for Large Knowledge Bases in Description Logics

Thi Le Pham, Nhan Le Thanh, Peter Sander

Decomposition-based Reasoning for Large Knowledge Bases in Description Logics

Thi Anh Le PHAM, Nhan LE-THANH and Peter SANDER *

Laboratoire I3S, Université de Nice Sophia-Antipolis, France
{tpham,nhan.le-thanh,peter.sander}@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 requirements are both significant factors that directly influence the performance of a reasoning algorithm. In this paper, we investigate a new technique for optimizing DL reasoning in order to minimize the above two factors 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 optimization techniques in current DL systems can effectively solve intractable inferences. Our technique is called "overlap ontology decomposition", in which the decomposition of a given ontology into many sub-ontologies is implemented such that the semantics and inference services of the original ontology are preserved. We are concerned about how to reason effectively with multiple KBs and how to improve the efficiency of reasoning over component ontologies.

1 Introduction

In recent years, much progress has been made in developing optimization techniques for reasoning in Description Logics [**?**]. The requirements of optimization are always derived from practical applications for treating large Knowledge Bases (ontologies) in DLs. The problem of effective reasoning with respect to a large TBox is a big challenge because of intractable inferences, generally due to General Concept Inclusions (GCIs) [**?**]. Each GCI causes a disjunctive expression to be added to the label of every node in the tree generated by the tableaux algorithm, and this leads to an exponential increase in the size of the search space to be explored by the $\sqcup$-rule. Consequently, even a small number of GCIs can considerably degrade the performance of a DL reasoner. Therefore obvious optimization techniques for reasoning often focus on reducing the number of GCIs in a terminology. For example, lazy unfolding and absorption [**?**] have proven to be particularly effective in dealing with terminologies. Our research into parallelizing tableaux algorithms falls within this goal.

A wide variety of information sources exist on the internet, and Serafini [**?**] introduced the integration of information from multiple sources. These sources can be partially connected using *mappings* or *linkings* in Distributed Description Logics

* Corresponding Author: Laboratoire I3S, UMR 6070, CNRS, Les Algorithmes, Bât. Euclide 2000, Route des Lucioles, BP121, 06903 Sophia-Antipolis, France; Email: peter.sander@unice.fr, Tel: +33 (0)4 9296 5160.

(DDL) [**?**, **?**], ε - connections [**?**] and *importings* in Package-based Description Logics [**?**, **?**]. We will concentrate on the problem of splitting a wide variety of sources into multiple small sources. Instead of reasoning on a given ontology, the tableaux algorithms are implemented in parallel on many sub-ontologies generated from the original ontology. We seek techniques which ensure that the answers derived from reasoning algorithms are still correct. As a result, we introduce the decomposition of an ontology into several sub-ontologies within the same domain as the first ontology. Our technique is called "overlap ontology decomposition" in which decomposed ontologies (called sub-ontologies) keep all of the primitive concepts, primitive roles and defined concept names of the original ontology. Only the set of axioms are divided into several subsets. Hence, the number of axioms in each decomposed ontology is reduced significantly from the original ontology. DDL is used for representing the system of decomposed ontologies.

In this paper, the actual algorithm for decomposing ontologies is not presented. We only focus on the most important properties of the ontology decomposition — preserving the syntax, semantics, and inferences of the original ontology in the sub-ontologies. The reasoning in the system of decomposed ontologies is implemented with two methods. The first is called the *distributed method*, in which decomposed ontologies are represented in a distributed system by distributed description logic and a distributed tableau algorithm is applied. The second, called the *parallel method*, is the normal tableaux algorithm. While many other optimization techniques are only valid for certain DLs (they were created while developing DL systems and are applicable to a large range of reasoning systems), our techniques are completely independent of the DL supported by the reasoners. The algorithms are designed so that reasoning results obtained within the decomposed ontologies are the same as for reasoning in the original ontology.

The paper is organized as follows. In Section 2 we discuss some related work, some basic notions in DL and DDL are recalled, and we also review several optimization techniques that are applied in current DL reasoning systems. Section 3 defines an overlap ontology decomposition of a TBox and presents a distributed TBox along with its properties. In Section 4, we describe the algorithms for parallel reasoning and distributed reasoning in DDL, a detailed proof of the inference preservation in sub-ontologies is presented, and we also propose the soundness and completeness of the reasoning algorithm in DDL, discussions about DDL and ontology decomposition, and parallel and distributed reasoning. Finally, we present our conclusions and future work in Section 5.

2 Related work

2.1 Description Logics

Description Logics (DLs) are a family of knowledge representation formalisms that allow presenting the knowledge of an application domain (the "world") in a structured and formally well-understood way. A knowledge base presented by a DL language is composed of two components, called *TBox* and *ABox*. The TBox introduces the vocabulary of an application domain (terminology) while the ABox contains the individuals appearing in the domain (the description of the world).

2.1.1 Syntax

In DL languages, the world is viewed as individuals that can be grouped into classes, called *concept names*, and their properties or attributes are binary relationships, called *role names*. For example, concept "GRADUATE-STUDENT" denotes a class of graduate students, the role "hasPub" denotes a relationship of pairs of persons (denoted by concept "PERSON") and publications (denoted by concept "PUBLICATION"). A particular DL provides a specific set of *constructors* (see Table **??**) that permits us to establish the more complex concept descriptions, i.e., the concept expressions that are constructed from *atomic concepts* and *atomic roles* using concept (role) constructors provided by a particular DL. For instance, concept constructors as conjunction (written $A \sqcap B$), universal quantification (written $\forall r.C$), and existential quantification (written $\exists R.C$) can be used to describe the object-oriented model classes with multiple super-classes and constraint types on members/attributes. For example, the expression $\exists hasChild.MALE$ describes the concept of "The set of individuals who have at least one male child". The more complex description

$$\text{PERSON}\sqcap\text{MALE}\sqcap(\leq 2\text{hasChild})\sqcap\forall\text{hasChild.FEMALE}$$

denotes "A man who has at most two children, all of whom are female".

As a result, the DL languages are identified by the letters associated with each set of constructors, see in Table **??**

Concept constructors	$\mathcal{EL}$	$\mathcal{ELN}$	$\mathcal{FL}^-$	$\mathcal{AL}$	$\mathcal{ALE}$	$\mathcal{ALC}$
universal ($\top$)	×	×		×	×	×
bottom ($\bot$)				×	×	×
conjunction	×	×	×	×	×	×
disjunction						×
negation				×	×	×
existential quantification	×	×	×	×	×	×
universal quantification			×	×	×	×
number restriction ($\leq$, $\geq$)		×				

Table 1. Some DL languages

A TBox contains a finite set of concept descriptions, called *terminological axioms* (or more briefly, just *axioms*). The axiom is an introduction of the names of new concepts and new roles, or an assertion of a subsumption relationship between the concepts and the roles that are functional or transitive. Given a DL language $\mathcal{L}$, an axiom in $\mathcal{L}$ has one of the forms:

1. $A \sqsubseteq C$, primitive concept specification
2. $A \equiv C$, concept definition
3. $C \sqsubseteq D$, general concept inclusion (GCI)
4. $C \equiv D$, concept equation

where A is a concept name, C, D are concept expressions in $\mathcal{L}$.

For example, we examine an ontology $\mathcal{T}_{uni}$ for describing a small part of a university and the activities that occur there (extracting some axioms from SHOE ontology [**?**]).

$\mathcal{T}_{uni}$

STUDENT	$\sqsubseteq$ PERSON
LECTURER	$\sqsubseteq$ PERSON
GRADUATE-STUDENT	$\sqsubseteq$ STUDENT
THESIS	$\sqsubseteq$ PUBLICATION
ARTICLE	$\sqsubseteq$ PUBLICATION
BOOK	$\sqsubseteq$ PUBLICATION
MASTER-THESIS	$\sqsubseteq$ THESIS
DOCTORAL-THESIS	$\sqsubseteq$ THESIS
DOCTOR	$\sqsubseteq$ GRADUATE-STUDENT $\sqcap\exists$ hasPub.DOCTORAL-THESIS
PROFESSOR	$\sqsubseteq$ DOCTOR$\sqcap\exists$ work-at.UNIVERSITY

Table 2. An example of TBox $\mathcal{T}_{uni}$

2.1.2 Semantics

In order to define the semantics of concept terms, one considers their *interpretations* $\mathcal{I}$, that are composed of a non empty set $\Delta^{\mathcal{I}}$ (the interpretation domain) and an interpretation function that assigns every atomic concept A to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and every role R to one binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of a concept C if $C^{\mathcal{I}}$ is not empty. A concept C is satisfiable if it has a model and unsatisfiable otherwise. We say that C is subsumed by D if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, for all $\mathcal{I}$; and C is equivalent to D ($C \equiv D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$, for every $\mathcal{I}$. For example, STUDENT$^{\mathcal{I}}$ = {Emile, Mary, Marc}; or hasPub$^{\mathcal{I}}$ = {(Marc, ThD1),(Marc, Ar1), (Frank, ThD1), (Frank, Ar2)}.

An interpretation $\mathcal{I}$ is a *model* of a TBox iff it satisfies every axiom $A \equiv C$ and $P \equiv R$ in the TBox, i.e., if $A^{\mathcal{I}} = C^{\mathcal{I}}$ and $P^{\mathcal{I}} = R^{\mathcal{I}}$ for all these definitions.
For *assertion* component (ABox) of a knowledge base, one can introduce the individuals (by giving names to them), and dominate their properties. If a, b are individual names, C is a concept term and R is a role term, then $C(a)$ and $R(a, b)$ are assertions. A finite set of such assertions is called an ABox $\mathcal{A}$. An interpretation $\mathcal{I}$ is a model of $\mathcal{A}$ iff $a^{\mathcal{I}} \in A^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

GRADUATE-STUDENT(Mary), GRADUATE-STUDENT(Frank), STUDENT(Mary),
DOCTORAL-THESIS(ThD1), DOCTORAL-THESIS(ThD2),
ARTICLE(Ar1), ARTICLE(Ar2),
hasPub(Marc, ThD1), hasPub(Frank, ThD2), hasPub(Frank, Ar2),
work-at(Frank, Nice-University)

Table 3. An example of ABox $\mathcal{A}_{uni}$

Constructors	Syntax	Semantic
top $\top$	$\top$	$\Delta^{\mathcal{I}}$
bottom $\bot$	$\bot$	$\emptyset$
conjunction	$C_1 \sqcap ... \sqcap C_n$	$C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}$
disjunction ($\mathcal{U}$)	$C_1 \sqcup ... \sqcup C_n$	$C_1^{\mathcal{I}} \cup ... \cup C_n^{\mathcal{I}}$
negation ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
existential quantification ($\mathcal{E}$)	$\exists R.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e : (d,e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
universal quantification	$\forall R.C$	$\{d \in \Delta^{\mathcal{I}} \mid \forall e : (d,e) \in R^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\}$
number restriction ($\mathcal{N}$) ($\geq$)	$\geq nR$	$\{d \in \Delta^{\mathcal{I}} \mid card(e \in \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}) \geq n\}$
number restriction ($\mathcal{N}$) ($\leq$)	$\leq nR$	$\{d \in \Delta^{\mathcal{I}} \mid card(e \in \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}) \leq n\}$
role conjunction ($\mathcal{R}$)	$R_1 \sqcap ... \sqcap R_n$	$R_1^{\mathcal{I}} \cap ... \cap R_n^{\mathcal{I}}$

Table 4. Syntax et Semantics of concept and role descriptions

2.1.3 Reasoning Services

The goal of a knowledge representation system is not only to store the concept definitions and the assertions but also to provide inference services which make it possible to obtain knowledge which is not represented explicitly in the base. For example, the assertion PROFESSOR(Frank) can be drawn from TBox $\mathcal{T}_{uni}$ (Table **??**) and ABox $\mathcal{A}_{uni}$ (Table **??**).

- *Subsumption*. The concept description D subsumes the concept description C w.r.t. TBox $\mathcal{T}$, written $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of TBox $\mathcal{T}$. For example, $STUDENT \sqsubseteq PERSON$. C, D are equivalent w.r.t TBox $\mathcal{T}$, written $C \equiv_{\mathcal{T}} D$ iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$
- *Satisfiability*. A concept description C is satisfiable (or consistent) w.r.t. TBox $\mathcal{T}$ iff there exist a model $\mathcal{I}$ of TBox $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. In this case, we say that $\mathcal{I}$ is a model of C.
- *Instance Checking*. An individual a is an instance of a concept description C w.r.t. TBox $\mathcal{T}$ and ABox $\mathcal{A}$, written $\langle \mathcal{T}, \mathcal{A} \rangle \models C(a)$, iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$

In fact, the satisfiability of concept description can be reduced to the subsumption problem and inversely: For two concept descriptions C, D, we have :

- C is subsumed by $D \Longleftrightarrow C \sqcap \neg D$ is unsatisfiable.
- C is unsatisfiable $\Longleftrightarrow C$ is subsumed by $\bot$.

2.1.4 Tableau Algorithm

The reasoning in DL is essentially based on first-order tableaux calculus. This technique eliminates redundant checks in the tableaux in order to give a strict upper bound on the complexity of the method. Checking the satisfiability of a concept description D is implemented by showing a model - an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ in which $D^{\mathcal{I}} \neq \emptyset$. Such a model is presented by a *completion tree* **T**, in which the nodes correspond to individuals, and edges are role-successor relationships (relationship between individuals in the interpretation of a role). Each node x in the tree is labeled with a set $\mathcal{L}(x)$ of primitive concepts and the negation of primitive concepts which must satisfy:

$$C \in \mathcal{L}(x) \Rightarrow x \in C^{\mathcal{I}}$$

Each edge $\langle x, y\rangle$ in the tree is labeled with the role name:

$$R = \mathcal{L}(\langle x, y\rangle) \Rightarrow \langle x, y\rangle \in R^{\mathcal{I}}$$

Assume that the concept description D is represented in $\mathcal{ALC}$ language. An $\mathcal{ALC}$-tableau must satisfy the following conditions:

- T1: If $C \in \mathcal{L}(x)$, then $\neg C \notin \mathcal{L}(x)$
- T2: If $C_1 \sqcap C_2 \in \mathcal{L}(x)$, then $C_1 \in \mathcal{L}(x)$ and $C_2 \in \mathcal{L}(x)$,
- T3: If $C_1 \sqcup C_2 \in \mathcal{L}(x)$, then $C_1 \in \mathcal{L}(x)$ or $C_2 \in \mathcal{L}(x)$,
- T4: If $\forall R.C \in \mathcal{L}(x)$ and $R \in \mathcal{L}(\langle x, y\rangle)$, then $C \in \mathcal{L}(y)$,
- T5: If $\exists R.C \in \mathcal{L}(x)$, then there is some y s.t. $R \in \mathcal{L}(\langle x, y\rangle)$ and $C \in \mathcal{L}(y)$.

The strategy of tableaux algorithms is internalization before implementing reasoning. Given axiom $i : E \sqsubseteq F$, its internalized concept is $i : C_{E\sqsubseteq F} = \neg E \sqcup F$; for TBox $\mathcal{T}_i$, its internalized concept is $C_{\mathcal{T}_i} = \bigcap_{E\sqsubseteq F \in \mathcal{T}_i} \neg E \sqcup F$. Also, the role hierarchy $\mathcal{R}_{\mathcal{T}_i}$ contains the role axioms of $\mathcal{T}_i$, plus additional axioms $P \sqsubseteq U$, for each role P of $\mathcal{T}_i$, with U some universal role.

The algorithm will start with the tree consisting of a singe node x_0 labeled with $\mathcal{L}(x_0) = \{C \sqcap C_{\mathcal{T}}\}$, where $C_{\mathcal{T}} = \sqcap_{(C_i \sqsubseteq D_i)\in\mathcal{T}} (\neg C_i \sqcup D_i)$ and $\{C_i \sqsubseteq D_i\}$ is the set of axioms in $\mathcal{T}$. **T** is then expanded from node x_0 by repeatedly applying the *expansion rules* from Table **??**. The algorithm terminates either when **T** is *complete*

$\sqcap$-rule	if	1. $(C_1 \sqcap C_2) \in \mathcal{L}(x)$
		2. $\{C_1, C_2\} \nsubseteq \mathcal{L}(x)$
	then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule	if	1. $(C_1 \sqcup C_2) \in \mathcal{L}(x)$
		2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
		3. $D = C_1$ or $D = C_2$
	then	a. $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{D\}$
$\exists$-rule	if	1. $\exists R.C \in \mathcal{L}(x)$
		2. there is no y such that $\mathcal{L}(\langle x, y\rangle) = R$ and $C \in \mathcal{L}(y)$
	then	create a new node y and edge $\langle x, y\rangle$
		with $\mathcal{L}(y) = \{C\}$ and $\mathcal{L}(\langle x, y\rangle) = R$
$\forall$-rule	if	1. $\forall R.C \in \mathcal{L}(x)$
		2. there is no some y such that $\mathcal{L}(\langle x, y\rangle) = R$ and $C \notin \mathcal{L}(y)$
	then	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$

Table 5. Expansion rules for $\mathcal{ALC}$ language

(no further expansion rules can be applied) or when it contains a *clash*. A node x of a completion tree **T** contains a clash if $\bot \in \mathcal{L}(x)$ or for some $A \in C, \{A, \neg A\} \subseteq \mathcal{L}(x)$. The algorithm is ensured of termination using a blocking strategy, where a node x is said to be *blocked* if there is some ancestor node y s.t. $\mathcal{L}(x) \subseteq \mathcal{L}(y)$, y is called the *blocking* node. For this approach, one can give the base of the decision algorithms for the satisfiability and subsumption which are sound and complete with the very expressive DL languages. However the investigation of the difference between the expressivity of DL languages and the complexity of their inference problems is still one of the most important questions in research on DLs.

2.2 Distributed Description Logics

Distributed Description Logics (DDL) is proposed by Borgida and Serafini [**?**] for representing and reasoning about knowledge bases (ontologies) in distributed environments. A knowledge base (ontology) built by a concept language is generally composed of two component levels: intensional (called TBox) and extensional (called ABox). In DDL, each ontology corresponds to a description logic theory (TBox) and ontologies are linked by semantic mappings (*bridge rules*). We briefly review some definitions of DDL as given in [**?**].

2.2.1 Syntax

Let $\{\mathcal{DL}_i\}_{i\in I}$ be a collection of description logics, where I is a non empty set of indices. For each $i \in I$, a TBox $\mathcal{T}_i$ is presented in a particular $\mathcal{DL}_i$. In order to distinguish descriptions in each TBox $\mathcal{T}_i$, we prefix the descriptions with the index of their TBoxes. For example, $i : C$ denotes a concept C of $\mathcal{DL}_i$. The semantic mappings between different TBoxes are depicted by using bridge rules.

Definition 1. *(Bridge rule) A bridge rule from $\mathcal{T}_i$ to $\mathcal{T}_j$ is an expression in one of the following three forms:*

$$i : x \xrightarrow{\sqsubseteq} j : y, \quad \textit{into-bridge rule}$$
$$i : x \xrightarrow{\sqsupseteq} j : y, \quad \textit{onto-bridge rule}$$
$$i : x \xrightarrow{\equiv} j : x, \quad \textit{identity-bridge rule}$$

where x and y are either two concepts, two roles, or two individuals of $\mathcal{DL}_i$ and $\mathcal{DL}_j$ respectively.

The bridge rules from $\mathcal{T}_i$ to $\mathcal{T}_j$ represent the relations between $\mathcal{T}_i$ and $\mathcal{T}_j$ from the j-th TBox point of view. In particular, the into-bridge rule $i : A \xrightarrow{\sqsubseteq} j : G$ expresses that, from the point of view of $\mathcal{T}_j$, the concept A in $\mathcal{T}_i$ is less general than its local concept G. Similarly with the onto-bridge and identity-bridge rules.

Definition 2. *(Distributed TBox) A distributed TBox (DTB) $\mathfrak{T} = \langle\{\mathcal{T}_i\}_{i\in I}, \mathcal{B}\rangle$ consists of a collection of TBoxes $\{\mathcal{T}_i\}_{i\in I}$ and a collection of bridge rules $\mathcal{B} = \{\mathcal{B}_{ij}\}_{i\neq j\in I}$ between them.*

2.2.2 Semantics

Each TBox $\mathcal{T}_i$ is interpreted by a local interpretation $\mathcal{I}_i$ in its local domain. Bridge rules are interpreted by using domain relation r_{ij} between domains.

Definition 3. *(Domain Relation) A domain relation r_{ij} from $\Delta^{\mathcal{I}_i}$ to $\Delta^{\mathcal{I}_j}$ is a subset of $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. We denote:*

$$r_{ij}(d) = \{d' \in \Delta^{\mathcal{I}_j} \mid (d, d') \in r_{ij}\}$$
$$r_{ij}(D) = \cup_{d\in D} r_{ij}(d)$$
$$r_{ij}(R) = \cup_{(d,d')\in R} r_{ij}(d) \times r_{ij}(d')$$

with $D \subseteq \Delta^{\mathcal{I}_i}$ and $R \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$.

The domain relation represents the possibility of mapping individuals from $\Delta^{\mathcal{I}_i}$ to $\Delta^{\mathcal{I}_j}$ from the point of view of $\mathcal{DL}_j$.

Definition 4. *(Distributed Interpretation) A distributed interpretation* $\mathfrak{I} = \langle \{\mathcal{I}_i\}_{i\in I}, \{r_{ij}\}_{i\neq j\in I}\rangle$ *of a DTB combines the local interpretations* $\mathcal{I}_i$ *of each* $\mathcal{T}_i$ *on the local domain* $\Delta^{\mathcal{I}_i}$ *and a family of relations* $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ *between local domains.*

A distributed interpretation $\mathfrak{I} = \langle(\{\mathcal{I}_i\}_{i\in I}, \{r_{ij}\}_{i\neq j\in I}\rangle$ is said to satisfy the elements of a DTB $\mathfrak{T} = \langle\{\mathcal{T}_i\}_{i\in I}, \mathcal{B}\rangle$ if:

$$
\begin{array}{ll}
\mathfrak{I} \models_d i : A \sqsubseteq \overrightarrow{\exists} j : G & \text{if } r_{ij}(A^{\mathcal{I}_i}) \subseteq G^{\mathcal{I}_j} \\
\mathfrak{I} \models_d i : B \overrightarrow{\sqsupseteq} j : H & \text{if } r_{ij}(B^{\mathcal{I}_i}) \supseteq H^{\mathcal{I}_j} \\
\mathfrak{I} \models_d i : A \sqsubseteq B & \text{if } \mathcal{I}_i \models A \sqsubseteq B \\
\mathfrak{I} \models_d \mathcal{T}_i & \text{if } \mathcal{I}_i \models \mathcal{T}_i \\
\mathfrak{I} \models_d \mathfrak{T} & \text{if } \mathfrak{I} \models_d \mathcal{T}_i \text{ and } \mathfrak{I} \models_d \mathcal{B}_{ij}, \forall i,j \in I \\
\mathfrak{T} \models_d i : A \sqsubseteq B & \text{if } \forall \mathfrak{J}, \mathfrak{J} \models_d \mathfrak{T} \Longrightarrow \mathfrak{J} \models_d i : A \sqsubseteq B
\end{array}
$$

Definition 5. *(Distributed Abox) A distributed Abox (DAB)* $\mathfrak{A} = \left\langle \{A_i\}_{i\neq j\in I}, \mathfrak{C} \right\rangle$ *consists of a set of Aboxes* $\{A_i\}_{i\in I}$*, and a set* $\mathfrak{C} = C_{ij}{}_{i\neq j\in I}$ *of partial, identity and complete individual correspondences from* i *to* j*. For every* $k \in I$*, all descriptions in* A_k *must be in the corresponding language* $\mathcal{DL}_k$*, and for every correspondence rule* $i : x \mapsto j : y$ *or* $i : x \overset{\equiv}{\mapsto} j : x$ *or* $i : x \overset{=}{\mapsto} j : \{y_1, y_2, ...\}$ *in* $\mathcal{C}_{ij}$*, the individual name* x *must be in* $\mathcal{DL}_i, \mathcal{DL}_j$*, and* $y_1, y_2, ...$ *must be in* $\mathcal{DL}_j$*.*

A distributed interpretation $\mathfrak{J}$ d-satisfies the elements of a DAB $\mathfrak{A} = \left\langle \{A_i\}_{i\in I}, \mathfrak{C} \right\rangle$ according to the following clauses: For every $i, j \in I$

$$
\begin{array}{rl}
\mathfrak{J} \models_d i : x \mapsto j : y, & \text{if } y^{\mathcal{I}_j} \in r_{ij}(x^{\mathcal{I}_i}) \\
\mathfrak{J} \models_d i : x \overset{\equiv}{\mapsto} j : x, & \text{if } r_{ij}(x^{\mathcal{I}_i}) = x^{\mathcal{I}_j} \\
\mathfrak{J} \models_d i : x \overset{=}{\mapsto} j : \{y_1, y_2, ...\}, & \text{if } r_{ij}(x^{\mathcal{I}_i}) = \{y_1^{\mathcal{I}_j}, y_2^{\mathcal{I}_j}, ...\} \\
\mathfrak{J} \models_d i : C(a), & \text{if } \mathcal{I}_i \models_d C(a) \\
\mathfrak{J} \models_d i : p(a,b), & \text{if } \mathcal{I}_i \models_d p(a,b) \\
\mathfrak{J} \models_d A_i, & \text{iff } \mathfrak{J} \models_d \pi \text{ for every assertion } \pi = C(a), p(a,b) \text{ in } A_i \\
\mathfrak{J} \models_d \mathfrak{A}, & \text{if for } \forall i \in I,\ \mathfrak{J} \models_d A_i \text{ and} \\
 & \quad \mathfrak{J} \text{ d-satisfies every individual correspondence in } \mathfrak{C}. \\
\mathfrak{A} \models_d i : C(a), & \text{if for every distributed interpretation } \mathfrak{J}, \\
 & \quad \mathfrak{J} \models_d \mathfrak{A} \text{ implies } \mathfrak{J} \models_d i : C(a). \\
\mathfrak{A} \models_d i : p(a,b), & \text{if for every distributed interpretation } \mathfrak{J}, \\
 & \quad \mathfrak{J} \models_d \mathfrak{A} \text{ implies } \mathfrak{J} \models_d i : p(a,b).
\end{array}
$$

2.3 Optimization Techniques

The normal tableaux algorithm is too slow to form a basis for a useful system of description logic. Therefore we studied and used some optimization processes that

improve the execution of the algorithm of the satisfiability test. In this section, we are going to recall some optimization techniques for solving the reasoning problems in DL which aim at eliminating redundant non-deterministic applications of expansion rules during satisfiability checking (see, e.g., [**?**]) used in systems as FACT [**?**].

2.3.1 Normalization, simplification and encoding

In general terminologies, large and complex concepts are usually built up from less complex descriptions. Whenever checking the satisfiability of a concept expression by the basic tableaux algorithm, i.e., unfolding these large and complex concepts, a clash is only detected when an atomic concept and its negation occur in the same node label. The algorithm is not effective when a clash can be detected immediately due to obvious unsatisfiability in its description.

Example 1. Testing the satisfiability of the concept expression $\exists R.C \sqcap \forall R.\neg(C \sqcup D)$ where R is a complex role and C, D are complex concepts.

If C is large, then the unfolding of C can lead to costly wasted work, while if $\exists R.C \sqcap \forall R.\neg(C \sqcup D)$ is transformed into $\sqcap\{\neg(\forall R.\neg C), \forall R.\neg\sqcap\{C, D\}\}$, then a contradiction will be detected immediately, independently of the structure of C. Hence, the detection of clashes can be addressed by normalizing and recoding all concept expressions into a lexically normalized form, and by identifying lexically equivalent expressions. These transformations are realized by the *normalize* and *encode* functions in [**?**]. Simplifications can also be examined during the normalizing process for eliminating redundancy and help to identify obvious satisfiability and unsatisfiability. Contradictions would be detected whenever a concept expression and its negation appear in the same node label.

2.3.2 GCI absorption

Absorption has been a key optimization technique in the past for processing DL ontologies [**?**]. It aims to reduce the number of GCIs by absorbing them into primitive concept introduction axioms whenever possible. The structure of absorbable GCIs is suggested by the syntax of the Grail concept description language in the Galen terminology that belongs to one of the following forms [**?**]:

1. GCI whose antecedent is only a primitive concept name.
2. GCI whose antecedent is either a conjunctive concept expression or a non-primitive concept name whose definition is a conjunctive concept expression. The first conjunct of a conjunctive concept expression is always either a primitive concept name or a non-primitive concept name whose definition is a conjunctive concept expression.

- For GCIs of form **??**, $CN \sqsubseteq C$ and $CN \sqsubseteq D$ are absorbed into :
  $CN \sqsubseteq C \sqcap D$
  with semantic correspondence:
  $CN^{\mathcal{I}} \subseteq C^{\mathcal{I}} \wedge CN^{\mathcal{I}} \subseteq D^{\mathcal{I}} \Longleftrightarrow CN^{\mathcal{I}} \subseteq C^{\mathcal{I}} \cap D^{\mathcal{I}}$

- For GCIs of form **??**: $CN \sqcap C \sqsubseteq D$ is absorbed into:
  $CN \sqsubseteq D \sqcup \neg C$
  with semantic correspondence:
  $CN^{\mathcal{I}} \cap C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \Longleftrightarrow CN^{\mathcal{I}} \subseteq D^{\mathcal{I}} \cup \neg C)^{\mathcal{I}}$
  where CN is a primitive concept name and C, D are concept expressions.

2.3.3 Semantic branching

Syntactic branching in standard tableaux algorithms works by choosing an unexpanded disjunction and searching the different models obtained by adding each of the disjuncts. These branches are not disjoint, so an unsatisfiable disjunct may occur in different branches and it would lead to wasted effort for discovering the unsatisfiability. This problem is dealt with by the semantic branching technique adapted from the Davis-Putnam-Logemann-Loveland procedure (DPL), for example tableau expansion of a node x, where $\{(C \sqcup D_1), (C \sqcup D_2)\} \subseteq \mathcal{L}(x)$ and C is an unsatisfiable concept expression. Syntactic branching will generate four branches in which two branches contain C — this is obviously redundant. By choosing only a single disjunct C and adding $\neg C$ to $\mathcal{L}(x)$, we obtain two possible sub-trees for searching by the semantic branching technique.

2.3.4 Heuristics

For reasoning algorithms, heuristics can be employed to attempt searching in some "good" order for inference rule applications (rule heuristics) and, for non-deterministic rules, the order in which to explore the different expansions chosen by rule applications (expansion-order heuristics). The goal is to try to select an order that leads earlier to a model discovery (the input will be satisfiable) or to a proof that no model exists (the input will be unsatisfiable). One of the most well-known and widely used heuristics in the SAT problem is MOMS (Maximum Occurrences in clause of Minimum Size) [**?**]. It is simple, easy to apply, rather precise, and problem-independent. Its goal in DL reasoning is simply to prefer the expressions having maximum occurrences of a disjunct in the disjunctions of minimum size.

3 Decomposition

As mentioned above, the optimization techniques presented are only actually effective in some cases with the particular structures of GCIs. Our work consists in eliminating GCIs as much as possible from the general ontology (a TBox) by decomposing an ontology into several sub-ontologies (a distributed TBox). The reasoning is then implemented on these sub-ontologies that have overlap in content. The reasoning effect depends on the decomposition method of the original TBox, and we propose a technique called "overlap ontology decomposition". In this paper, we examine only the simplest case, decomposing a TBox into two smaller TBoxes. The general case is presented in [**?**].

3.1 Definitions

Let $\mathbf{C}, \mathbf{R}, \mathbf{B}, \mathbf{A}$ be the sets of primitive concepts, primitive roles, defined concept names, and axioms respectively of TBox $\mathcal{T}$. We denote:

$$\mathcal{T} = (\mathbf{C}, \mathbf{R}, \mathbf{B}, \mathbf{A}),$$
$$\mathbf{B} = \{B \mid B \equiv C)\},$$
$$\mathbf{A} = \{(D \sqsubseteq E)\},$$

where B is a concept name and C, D, E are concept expressions.

Note that $B \equiv C$ is a particular form of $D \sqsubseteq E$ because $(B \equiv C) \Longleftrightarrow (B \sqsubseteq C \text{ and } \mathrm{C} \sqsubseteq \mathrm{B})$, however $B \equiv C$ is also called a defined concept. Thus, a concept definition is also an axiom. We define a decomposition of $\mathcal{T}$ called *overlap decomposition* as follows:

Definition 6. *(Overlap Decomposition)* $\langle \{\mathcal{T}_i\}, \{\mathcal{B}_{ij}\}_{i \neq j \in \{1,2\}} \rangle$ *(denoted by* $\mathfrak{T}$ *or* $\mathcal{T}_{12}$*) is an overlap decomposition of a TBox* $\mathcal{T}$ *if:*

- *Each component* $\mathcal{T}_i$ *consists of all primitive concepts, primitive roles and concept names of* $\mathcal{T}$*, i.e.,* $\mathcal{T}_i = (\mathbf{C}, \mathbf{R}, \mathbf{B}, \mathbf{A}_i)$*, where* $\mathbf{A}_i = \{(i : C \sqsubseteq i : D)\}$*, with* $i \in \{1, 2\}$.
- $\mathbf{A}_i \subseteq \mathbf{A}$; $\cup_i \mathbf{A}_i = \mathbf{A}$ *and* $\cap_i \mathbf{A}_i = \emptyset$.
- $\mathcal{B}_{ij}$ *is a set of semantic mappings (identity bridge rules) between* $\mathcal{T}_i$ *and* $\mathcal{T}_j$*. A semantic mapping is an identity relationship between two concepts (two roles) that co-occur in two axioms of different TBoxes (*$\mathcal{B}_{ij}$ *can be empty if* $\mathbf{A}_i$ *and* $\mathbf{A}_j$ *are "disjoint" [2]).*

Thus, an overlap decomposition of a TBox $\mathcal{T}$ is a special distributed TBox, because $\mathcal{B}_{ij}$ includes only identity bridge rules. It captures all properties of general distributed TBoxes (as given the section above) and some particular properties (see the following section). We need to give a strategy for decomposing $\mathcal{T}$ into subsets $\{\mathcal{T}_i\}$ such that each $\mathcal{T}_i$ is a DL formalism of a certain ontology.

The set of semantic mappings between two TBoxes is determined by finding concept (role) names that co-occur in these TBoxes, and by representing them in the form of bridge rules in DDL. We use $\mathcal{B}$ to indicate the whole set of bridge rules obtained.

In decomposition, the domain relation r_{ij} between $\mathcal{I}_i$ and $\mathcal{I}_j$ can be considered as an identity mapping in that it maps an object of $\Delta^{\mathcal{I}_i}$ to itself in $\Delta^{\mathcal{I}_j}$, i.e., $r_{ij}(x) = x$. The essence of this decomposition is to only examine the axioms, i.e., the set of axioms of the original TBox are divided into several subsets of axioms for sub-TBoxes. We are concerned with the axioms because their presence can lead to an ExpTime lower bound for the complexity of the reasoning problem [**?**]. However, the large number of concepts and roles presents difficulties for maintaining, reusing and developing independently of the component ontologies.

For example, to illustrate a decomposition of TBox $\mathcal{T}_{uni}$ in Table **??**, see Table **??**. Table **??** depicts two decomposed TBoxes $\mathcal{T}_1, \mathcal{T}_2$ of TBox $\mathcal{T}_{uni}$. $\mathcal{T}_1, \mathcal{T}_2$ have a semantic mapping

[2] $\mathbf{A}_i$ and $\mathbf{A}_j$ are called disjoint if there is not a common concept (a role) occurring in both $\mathbf{A}_i$ and $\mathbf{A}_j$

$\mathcal{T}_1$

STUDENT	$\sqsubseteq$ PERSON
LECTURER	$\sqsubseteq$ PERSON
GRADUATE-STUDENT	$\sqsubseteq$ STUDENT
DOCTOR	$\sqsubseteq$ GRADUATE-STUDENT $\sqcap\exists$ hasPub.DOCTORAL-THESIS
PROFESSOR	$\sqsubseteq$ DOCTOR $\sqcap\exists$ work-at.UNIVERSITY

$\mathcal{T}_2$

THESIS	$\sqsubseteq$ PUBLICATION
ARTICLE	$\sqsubseteq$ PUBLICATION
BOOK	$\sqsubseteq$ PUBLICATION
MASTER-THESIS	$\sqsubseteq$ THESIS
DOCTORAL-THESIS	$\sqsubseteq$ THESIS

Table 6. A decomposition of TBox $\mathcal{T}_{uni}$ into its two sub-TBoxes

1:DOCTORAL-THESIS $\underset{\rightarrow}{\equiv}$ 2:DOCTORAL-THESIS, where DOCTORAL-THESIS is a concept name. We only show the decomposition of the set of axioms, in addition $\mathcal{T}_{uni}, \mathcal{T}_1$ and $\mathcal{T}_2$ have a shared set of concepts and roles :
{STUDENT, PERSON, LECTURER, GRADUATE-STUDENT, THESIS, PUBLICATION, ARTICLE, BOOK, MASTER-THESIS, DOCTORAL-THESIS, DOCTOR, PROFESSOR, hasPub, work-at}.

3.2 Additive properties of distributed TBox in a decomposition

Decomposing a TBox is also represented by a distributed TBox, therefore it assumes all the properties of a general distributed TBox, as well as some additional properties. Concerning interpretations "satisfying" TBoxes, we can add :

$$\begin{array}{ll} \mathfrak{J} \models_d i : A \underset{\rightarrow}{\equiv} j : A & \text{if} \quad r_{ij}(A^{\mathcal{I}_i}) \equiv A^{\mathcal{I}_j} \\ \mathfrak{J} \models_d i : A \sqsubseteq j : B & \text{if} \quad r_{ij}(A^{\mathcal{I}_i}) \subseteq B^{\mathcal{I}_j} \\ \mathfrak{T} \models_d i : A \sqsubseteq j : B & \text{if} \quad \text{for } \forall \mathfrak{J},\ \mathfrak{J} \models_d \mathfrak{T} \Rightarrow\ \mathfrak{J} \models_d i : A \underset{\rightarrow}{\sqsubseteq} j : B \end{array}$$

Note that, all $\mathcal{T}_i, (i \in I)$ have the same set of concepts and roles of $\mathcal{T}$, so we write $i : X$ to show that we are talking about concept (role) X in $\mathcal{T}_i$. Furthermore, we particularly note that DDL expresses "directionality" / "no backflow", i.e., the set of bridge rules $\mathcal{B}_{ij}$ implies that information flow is only propagated from $\mathcal{T}_i$ to $\mathcal{T}_j$, and not in the inverse direction. This property is used to avoid the infinite loop when propagating tableaux algorithms between local ontologies, hence we need to designate the direction of bridge rules for each pair of TBoxes. For convenience, we stipulate that if there is a set of bridge rules $\mathcal{B}_{ij}$ from $\mathcal{T}_i$ to $\mathcal{T}_j$ then $\mathcal{T}_i$ is called the *source* TBox and $\mathcal{T}_j$ is called the *target* TBox.

3.3 Decomposition Properties

Intuitively, a decomposition $\langle\{\mathcal{T}_i\}, \mathcal{B}_{ij}\rangle$ of $\mathcal{T}$ is "good" if the quantity of common objects (concepts and roles) in the decomposed ontologies is as small as possible and

there are about the same number of axioms in each of the two sub-ontologies. Decomposed ontologies are consistent, in that they have the same domain as the original ontology, because they duplicate all concepts and roles of the original ontology. We can easily infer that the decomposed ontologies also have the same interpretation as the original ontology.

In addition, we have the following properties:

1. The preservation of concepts (roles): each concept (role) of $\mathcal{T}$ is also a concept (role) in all $\mathcal{T}_i$.
2. The preservation of axioms: $\mathbf{A} \subseteq \cup_i \mathbf{A}_i$. Indeed, each axiom $C \sqsubseteq D$ of ontology $\mathcal{T}$ is also an axiom in one of $\mathcal{T}_i$. Note that there is no axiom which co-occurs in two different ontologies (i.e., the sets of axioms of two different ontologies are disjoint).
3. The semantic mappings are proposed in order to conserve the relationships of entities between sub-ontologies, they present only identity relationships between two concepts (two roles) that co-occur in two axioms of two different ontologies. They are presented by bridge rules $\mathcal{B}$.
4. The preservation of semantics, i.e., $\Delta = \Delta_i$, and $\cdot^{\mathcal{I}} = \cdot^{\mathcal{I}_i}, i = 1, 2; \mathcal{I}, \mathcal{I}_i$ are the interpretations of $\mathcal{T}$ and $\mathcal{T}_i$ respectively.
5. The preservation of inference services, i.e., if a concept description is satisfiable w.r.t. $\mathcal{T}$, then it is also satisfiable w.r.t. $\langle \mathcal{T}_i, \mathcal{B} \rangle$, and conversely.

The three first properties are easy to deduce from the definition of decomposition, and prove the preservation of syntax. They ensure that ($\{\mathcal{T}_i\}$,$\mathcal{B}$) is well represented by DDL. In the following sections, we focus on the two most important properties of decomposition - semantic and inference preservation. The definitions and proofs are only examined in a DTB decomposition $\langle \{\mathcal{T}_i\}, \mathcal{B} \rangle$ of $\mathcal{T}$, i.e., $\mathcal{T}$ and $\{\mathcal{T}_i\}$ have the same set of concept and role names. Therefore, instead of saying "common (primitive) concept (role) names" we use the shorter "(primitive) concept (role) names".

Also from these properties, we propose two reasoning approaches with decomposed ontologies. By characterizing the decomposition form, we will show that there exists a deterministic algorithm making it possible to decompose an ontology represented by a DL into several ontologies represented by DDL conforming to this form.

Definition 7. *(Expansion) Formally, expanding a concept, a concept description, or even a bridge rule is defined recursively as follows:*
Expansion $Ex(.)$ is a mapping from a concept (a concept description, or a bridge rule) into a set of primitive concepts and primitive roles which occur in their definitions:

- *If A is an atomic primitive concept (role) or its negation, then* $Ex(A) = \{A\}$
- *If A is a defined concept in form* $\exists R.C$ *or* $\forall R.C$*, then* $Ex(A) = \{A\} \cup Ex(R) \cup Ex(C)$
- *If A is a defined concept in form* $C_1 \sqcap C_2$ *or* $C_1 \sqcup C_2$ *, then* $Ex(A) = \{A\} \cup Ex(C_1) \cup Ex(C_2)$
- *If A is a composite concept description in form* $\rho(M_1, ..., M_k)$*,where* ρ *is a concept constructor,* $Ex(\rho(M_1, ..., M_k)) = \cup_i \{Ex(M_i)\}$
- *For an into (onto) bridge rule* $B_k = 1 : C \stackrel{\sqsubseteq}{\Rightarrow} 2 : D$ ($B_k = 1 : C \stackrel{\sqsubseteq}{\Rightarrow} 2 : D$), $Ex(B_k) = Ex(C) \cup Ex(D)$
- *For an identity bridge rule* $B_k = 1 : C \stackrel{\equiv}{\Rightarrow} 2 : C$, $Ex(B_k) = Ex(C)$

- $Ex(.)$ *can be extended for a set of bridge rules:*

$$Ex(\{B_k\}) = \bigcup_k Ex(B_k), \textit{with } B_k \in \mathcal{B}_{ij}$$

Expanding a concept description C can be understood as the set of concepts and roles constructing C.

Definition 8. *(Decomposition Function) We define a decomposition function $\#()$ (as given in [**?**]) from concepts and roles of the original Tbox $\mathcal{T}$ to concepts and roles in $\mathcal{T}_i$ ($i \in I$) as follows:*

1. *If M is a primitive concept (role), then $\#(M) = (i : M)$*
2. *If M is a concept expression with concept constructor ρ taking k arguments, $M = \rho(M_1, ..., M_k)$,*
   then $\#(\rho(M_1, \ldots, M_k) = Top \sqcap \rho(\#(M_1), \ldots, \#(M_k))$
3. *If M is a role expression with role constructor ρ taking k arguments, $M = \rho(M_1, ..., M_k)$,*
   then $\#(\rho(M_1, \ldots, M_k) = \rho(\#(M_1), \ldots, \#(M_k))$

Example 2. $\#(Student \sqcap \exists hasPub.Article)$ produces
$Top \sqcap i : Student \sqcap \exists(i : hasPub).(Top \sqcap i : Article)$

We now provide a DTB in DDL. First, ANYTHING, NOTHING denote the top and bottom concepts of $\mathcal{T}$ respectively. They are distinguished from the top (Top$_i$) and bottom (Bot$_i$) concepts of $\mathcal{T}_i$.

We apply $\#$ for a global TB $\mathcal{T}$, then generate a DTB $\#(\mathcal{T}) = (\{\mathcal{T}_i\}_{i \in I}, \mathcal{B}_{ij})$ in the language DDL, where $\mathcal{B}_{ij}$ is the set of identity bridge rules, composed of the following axioms:

1. copy all the axioms of global TBox to the same local TBox $\mathcal{T}_i, i \in I$:
   $i : \#(X) \sqsubseteq \#(Y)$, for all $X \sqsubseteq Y \in \mathcal{T}$ and $\#(X) \in \mathcal{T}_i$; $\#(Y) \in \mathcal{T}_i$
2. $i : \#(X) \underset{\rightarrow}{\equiv} j : \#(X)$, for all $X \in \mathcal{T}$, $\#(X) \in \mathcal{T}_i$; $\#(X) \in \mathcal{T}_j$ and $X \in Ex(\mathbf{A}_i), X \in Ex(\mathbf{A}_2)$
3. $ANYTHING \sqsubseteq Top_i$ to ensure that Top$_i$ is not empty
4. $Bot_i \sqsubseteq NOTHING$ to restrict Bot$_i$ to always be the incoherent concept
5. $(i, A) \sqsubseteq Top_i$ to ensure that Top$_i$ is the proper local top of its IS-A hierarchy
6. to ensure that each role R is in the same domain and space of Top_i:
   $Top_i \sqsubseteq \forall(i, R).(Top_i)$ for every role R of $\mathcal{T}$ the space of (i, R) is in $\Delta^{\mathcal{I}_i}$
   $\exists(i, R).ANYTHING \sqsubseteq Top_i$, R is only defined in $\Delta^{\mathcal{I}_i}$.

Definition 9. *Given two Tboxes $\mathcal{T}_i$ and $\mathcal{T}_j, i \neq j$, with their interpretations $\mathcal{I}_i = (\Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i})$ and $\mathcal{I}_j = (\Delta^{\mathcal{I}_j}, \cdot^{\mathcal{I}_j})$ respectively. We say that $\mathcal{T}_i$ and $\mathcal{T}_j$ are interpreted in the same domain (or $\mathcal{I}_i$ and $\mathcal{I}_j$ have the same domain) iff $\Delta^{\mathcal{I}_i} = \Delta^{\mathcal{I}_j}$ and $\cdot^{\mathcal{I}_i} = \cdot^{\mathcal{I}_j}$ for all common primitive concepts (roles).*

As already mentioned in [**?**] by Borgida and Serafini, the semantics of concept descriptions denoted $\rho(\arg_1, .., \arg_n)$ is a function f_ρ of its arguments. We examine some examples:

Example 3. $(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ is equal to $f_\cap(C_1^{\mathcal{I}}, C_2^{\mathcal{I}})$ for an associated *semantic function* $f_\cap(XC_1, XC_2, \Delta^{\mathcal{I}}) = \{d \in \Delta^{\mathcal{I}} | XC_1(d) \subseteq XC_2\}$.

Example 4. $(\forall R.C)^{\mathcal{I}} = all(R,C)^{\mathcal{I}}$ is equal to $f_{all}(R^{\mathcal{I}}, C^{\mathcal{I}})$ for an associated *semantic function* $f_{all}(XR, XC, \Delta^{\mathcal{I}}) = \{d \in \Delta^{\mathcal{I}} | XR(d) \subseteq XC\}$.

Definition 10. *Let ρ be a concept constructor whose semantics can be expressed as $\rho(\arg_1, .., \arg_n)^{\mathcal{I}} = f_\rho(\arg_1^{\mathcal{I}}, ..., \arg_n^{\mathcal{I}}, \Delta^{\mathcal{I}})$, for a function $f_\rho(X_1, ..., X_n, DY)$ whose definition contains no references to $\mathcal{I}$. Let $B_1, ..., B_n, W, \Delta$ be sets such that $W \subseteq \Delta$, and each B_j is either a subset of W or of $W \times W, 1 \leq j \leq n$. Then ρ is called a local constructor if f_ρ satisfies : $f_\rho(B_1, ..., B_n, \Delta) = f_\rho(B_1, ..., B_n, W)$, when it is a concept or role constructor.*

Proposition 1. *If two Tboxes $\mathcal{T}_i$ and $\mathcal{T}_j$ are interpreted in the same domain, then $C^{\mathcal{I}_i} = C^{\mathcal{I}_j}$, for every C is an arbitrary common concept description of T_i and T_j.*

Proof. Since $\mathcal{T}_i$ and $\mathcal{T}_j$ are interpreted in the same domain, we have $C^{\mathcal{I}_j} \equiv C^{\mathcal{I}_i}$, $R^{\mathcal{I}_j} \equiv R^{\mathcal{I}_i}$, where C, R are a common primitive concept and role respectively. By induction on the structure of an arbitrary concept (role) B, we easily show that $B^{\mathcal{I}_j} \equiv B^{\mathcal{I}_i}$.

For arbitrary common concept descriptions M, they will be constructed from the common primitive concepts (roles). Suppose M has the form $\rho(M_1, ..., M_k)$, where concepts $M_i, ..., M_k$ are less complex than M, and occur also in $\mathcal{T}_j$, ρ is a concept constructor taking k arguments.

By structural induction on M, we obtain $M_p^{\mathcal{I}_j} \equiv M_p^{\mathcal{I}_i}\ \forall p = 1, .., k$. We need now show that $M^{\mathcal{I}_i} = M^{\mathcal{I}_j}$. Indeed, $M^{\mathcal{I}_j} \equiv (\rho(M_1, ..., M_k))^{\mathcal{I}_j} \equiv f_\rho(M_1^{\mathcal{I}_j}, ..., M_k^{\mathcal{I}_j}, \Delta^{\mathcal{I}_j}) \equiv f_\rho(M_1^{\mathcal{I}_i}, ..., M_k^{\mathcal{I}_i}, \Delta^{\mathcal{I}_i}) \equiv (\rho(M_1, ..., M_k))^{\mathcal{I}_i} \equiv M^{\mathcal{I}_i}$.

4 Reasoning with decomposed ontologies

4.1 Decomposition TBox

Reasoning within a large ontology can be reduced to some reasoning procedures in its sub-ontologies based on parallel or distributed approaches. Reasoning services in modern DLs are essential concept satisfiability and subsumption problems that are solved by tableaux algorithms. To simplify the description, we examine the ontology in language $\mathcal{ALC}$.

An $\mathcal{ALC}$ TBox decomposition (a distributed $\mathcal{ALC}$ terminology) is presented with the following formation rules:

- Axioms are of the form:

  $$i : C \sqsubseteq D | i : C \equiv D | i : R \sqsubseteq S$$

  where C and D are concept expressions in $\mathcal{T}_i$, R and S are role names in $\mathcal{T}_i$, and $i \in I$
- Concept expressions are of the form:

  $$i : CN | \top\ | \bot\ | \neg i : C\ | i : C \sqcap D\ | i : C \sqcup D | \exists R.i : C | \forall R.i : C$$

  where CN is a concept name in $\mathcal{T}_i$, C and D are concept expressions in $\mathcal{T}_i$, R and S are role names.

- Bridge rules are of the form:

$$i : X \underset{\rightarrow}{\equiv} j : X$$

for all $X \in \mathcal{T}$, $X \in \mathcal{T}_i$, $X \in \mathcal{T}_j$ and $X \in Ex(\mathbf{A}_i)$, $X \in Ex(\mathbf{A}_j)$

The semantics of distributed $\mathcal{ALC}$ concept expressions are described in terms of a global interpretation $\Im = \langle \{\mathcal{I}_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I} \rangle$ that needs to satisfy all the properties in Definition 4 and additive properties of decomposition TBox.

In the following sections, two approaches for reasoning in decomposed ontologies will be introduced in more detail, the so-called parallel and distributed techniques. The tableaux algorithms will be applied on local ontologies and then either merged in the parallel case or propagated in the distributed case.

4.2 Parallel Reasoning

The main idea of parallel reasoning for a decomposed TBox $\mathfrak{T} = \langle \{\mathcal{T}_i\}, \{\mathcal{B}_{ij}\} \rangle$ is to construct multiple local tableaux $\mathbf{T}_i$ on $\mathcal{T}_i$ instead of a single global tableau. The role of local TBoxes is the same in this approach, i.e., we don't need to distinguish source TBox and target TBox, because the tableaux algorithms are completely independently implemented on local TBoxes, and are then combined. In a previous version of this paper [**?**], we proposed merging all nodes of local tableaux. Nevertheless, in the worst case, i.e., all branches in local tableaux of $\mathcal{T}_i$ and $\mathcal{T}_j$ are open, merging all the nodes in $\mathbf{T}_i$ and $\mathbf{T}_j$ can be conducted in exponential time. To eliminate redundant merging, in this section we address a merging condition for nodes in the different tableaux.

First, we introduce a new notion, *complete bridge rule*, which is an identity relation $i : X \overset{\equiv}{\rightarrow} j : X$ with $X \in \{Ex(Q) \cup Ex(\mathcal{B}_{ij})\}$, where Q is a query expression. In the case that we are examining, Q is testing concept C. In other words, the set of complete bridge rules, denoted by $\mathcal{M}_{ij}$, is extended from $\mathcal{B}_{ij}$. It consists of a fixed component ($\mathcal{B}_{ij}$) and a variable component depending on the queries. It helps merging (in the parallel case) and propagation (in the distributed case) of local tableaux. Two nodes in two tableaux are only merged if there is at least one *complete bridge rule* between them. A combined tableau of local tableaux is defined as follows:

Definition 11. *(Combined Tableau) A combined tableau for a distributed TBox* $\mathfrak{T} = \langle \{\mathcal{T}_i\}, \{\mathcal{B}_{ij}\} \rangle$ *is a tuple* $\langle \{\boldsymbol{T}_i\}, \{m_{ij}\}_{i \neq j} \rangle$*, where* $\boldsymbol{T}_i$ *is a local tableau of* $\mathcal{T}_i$*,* m_{ij} *is the image relation between* $\boldsymbol{T}_i$ *and* $\boldsymbol{T}_j$*, such that it creates mappings from nodes in* $\boldsymbol{T}_i$ *to nodes in* $\boldsymbol{T}_j$*. For a node* $x \in \boldsymbol{T}_i$ *and a node* $y \in \boldsymbol{T}_j$*,* $(x, y) \in m_{ij}$ *if they contain at least a complete bridge rule.*

We concretely examine the satisfiability testing of a concept C w.r.t $\mathcal{T}$ and w.r.t $(\{\mathcal{T}_1, \mathcal{T}_2\})$. In order to test the satisfiability of concept C w.r.t $(\{\mathcal{T}_1, \mathcal{T}_2\})$, we implement simultaneously and independently normal tableaux algorithms $\mathbf{T}_1(C)$ and $\mathbf{T}_2(C)$ on the TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively. This results in:

1. If $\mathbf{T}_1(C)$ or $\mathbf{T}_2(C)$ is unsatisfiable then we conclude that C is also unsatisfiable w.r.t $\mathcal{T}$.
2. Else, i.e., all $\mathbf{T}_1(C)$ and $\mathbf{T}_2(C)$ are thus satisfiable then we make a merge of $\mathbf{T}_1$ and $\mathbf{T}_2$, denote $\mathbf{T}_1 \oplus \mathbf{T}_2$, as follows:
   After calculating tableaux on $\mathcal{T}_1$and $\mathcal{T}_2$, two model trees AM_1 and AM_2 are respectively generated. Non-clash nodes in AM_1 can be joined with non-clash nodes

in AM_2 by pairs (if they contain complete bridge rule), generating new nodes in a merging model tree.
Suppose that m non-clash leaf nodes of AM_1 are merged with n non-clash leaf nodes of AM_2 (m, n are non-negative integers), then $m \times n$ new nodes will be generated in the merging tree. If all these $m \times n$ nodes in the merging tree are also clashes, then we conclude that C is unsatisfiable w.r.t $\mathcal{T}_1, \mathcal{T}_2$. Otherwise we conclude that C is satisfiable, i.e., if there exists at least one node that is non-clash, then we conclude that C is satisfiable.

Finally, merging two nodes x_{T_1}-label $\mathcal{L}(x^C_{T_1})$ and x_{T_2}-label $\mathcal{L}(x^C_{T_2})$ into a single node $x_{T_{12}}$-label $\mathcal{L}(x^C_{T_{12}}) = \mathcal{L}(x^C_{T_1}) \cup \mathcal{L}(x^C_{T_2})$, according to the following principles:

- $\exists R.C_1 \in \mathcal{L}(x^C_{T_1})$ and $\exists R.C_1 \in \mathcal{L}(x^C_{T_2})$, then the union of these labels consists of a node $x_{T_{12}}$-label $\mathcal{L}(x^C_{T_{12}}) = \mathcal{L}(x^C_{T_1}) \cup \mathcal{L}(x^C_{T_2})$, and an edge-label R between $x_{T_{12}}$ and a node $y : \{C_1\}$
- $\exists R.C_1 \in \mathcal{L}(x^C_{T_1})$ and $\exists R.C_2 \in \mathcal{L}(x^C_{T_2})$, then the union of these labels is composed of $x_{T_{12}}$-label $\mathcal{L}(x^C_{T_{12}}) = \mathcal{L}(x^C_{T_1}) \cup \mathcal{L}(x^C_{T_2})$, and an edge-label R between $x_{T_{12}}$ and a node $y : \{C_1, C_2\}$
- $\exists R_1.C_1 \in \mathcal{L}(x^C_{T_1})$ and $\exists R_2.C_2 \in \mathcal{L}(x^C_{T_2})$, then the union of these labels consists of a node $x_{T_{12}}$-label $\mathcal{L}(x^C_{T_{12}}) = \mathcal{L}(x^C_{T_1}) \cup \mathcal{L}(x^C_{T_2})$, and two edges that one-label R_1 between $x_{T_{12}}$ and a node $y_1 : \{C_1\}$,and other-label R_2 between $x_{T_{12}}$ and a node $y_2 : \{C_2\}$
- $\exists R.C_1 \in \mathcal{L}(x^C_{T_1})$, $\forall S.C_2 \in \mathcal{L}(x^C_{T_2})$ and $R \sqsubseteq S$, then the union of these nodes is composed of node $x_{T_{12}}$-label $\mathcal{L}(x^C_{T_{12}}) = \mathcal{L}(x^C_{T_1}) \cup \mathcal{L}(x^C_{T_2})$, and an edge-label R between $x_{T_{12}}$ and a node $y : \{C_1, C_2\}$

where R, R_1, R_2 and S are role names, C_1 and C_2 are concept names.

Example 5. Merging two nodes x_1 with label $\mathcal{L}(x_1) = \{C, D, \exists R.C_2, \forall R.C_3\}$ and x_2 with label $\mathcal{L}(x_2) = \{C, \exists R.C_4, \forall R.C_3, \exists R_1.C_5\}$ of trees $\mathbf{T}_1$ and $\mathbf{T}_2$ respectively (see Figure **??**).

In parallel reasoning, tableaux algorithms are applied on local TBoxes, and termination is thus ensured by the blocking strategy [**?**].

However, this reasoning approach proves difficult in the merging implementation. In the case that the number of open nodes in local tableaux is large, merging can result in exponential behavior. So, instead of merging local tableaux, we propose using distributed reasoning in the following section.

4.3 Reasoning in Distributed Description Logics

In this section, we introduce a reasoning algorithm based on the distributed reasoning procedure of Serafini and Tamilin [**?**], which takes a complex concept C as input and returns the result of its (un)satisfiability test. In this approach, the reasoning needs to distinguish source (denoted by $\mathcal{T}_s$) TBoxes and target TBoxes (denoted by $\mathcal{T}_t$). To emphasize this difference, we denote a decomposed TBox by $\mathfrak{T} = \langle \{\mathcal{T}_s, \mathcal{T}_t\}, \mathcal{B}_{st} \rangle$ instead of $\mathfrak{T} = \langle \{\mathcal{T}_i\}, \mathcal{B}_{ij} \rangle$. For simplification, we only examine the distributed TBoxes that

Input: Concept C
Output: Satisfiable / Unsatisfiable

1: BEGIN
2: $\mathbf{T}_1 = Tab_1(C)$;
3: $\mathbf{T}_2 = Tab_2(C)$;
4: $M_{ij} = Ex(\mathcal{B}_{ij}) \cup Ex(C)$;
5: **if** ($\mathbf{T}_1$ is not clashed) and ($\mathbf{T}_2$ is not clashed) **then**
6: $\mathbf{T} = \emptyset$;
7: **for each** open branch β_1 in $\mathbf{T}_1$ **do**
8: **for each** open branch β_2 in $\mathbf{T}_2$ **do**
9: repeat
10: select node $x_1 \in \beta_1$ and node $x_2 \in \beta_2$
11: if $\mathcal{L}_1(x_1) \cap \mathcal{L}_2(x_2) \subset M_{ij}$ then
12: $\mathcal{L}_{12}(x_{12}) = \mathcal{L}_1(x_1) \cup \mathcal{L}_2(x_2)$
13: $\mathbf{T} = \mathbf{T} \cup x_{12}$
14: end if
15: until ((β_1 is open) and (β_2 is open) and (there exist not verified nodes in β_1 or β_2))
16: **end for**
17: **end for**
18: **end if**
19: **if** (($\mathbf{T}_1$ is clashed) or ($\mathbf{T}_2$ is clashed) or ($\mathbf{T}$ is clashed)) then
20: return unsatisfiable
21: **Else**
22: return satisfiable
23: END.

Table 7. Parallel tableaux algorithm

consist of one source TBox and one target TBox.
For example, checking the satisfiability of an $\mathcal{ALC}$ concept expression C w.r.t $\mathcal{T}$ and w.r.t $\mathfrak{T} = \langle\{\mathcal{T}_s, \mathcal{T}_t\}, \mathcal{B}_{st}\rangle$, where $\mathcal{B}_{st}$ is a set of semantic mappings (identity bridge rules) from $\mathcal{T}_s$ to $\mathcal{T}_t$.

The key idea is to first build a local completion tree $\mathbf{T}_t$ by running a local tableau algorithm on the target TBox $\mathcal{T}_t$. According to the tableau algorithm, each node x generated during completion tree building is labeled with a function $\mathcal{L}(x)$ containing concepts that x must satisfy. We then try to close open branches of $\mathbf{T}_t$ by checking the complete bridge rules between $\mathcal{T}_t$ and the other source TBoxes $\mathcal{T}_s$, which could pass the computation to those TBoxes. We note that the propagation of DTab only follows one direction, from s to t, and not in the opposite direction.

Definition 12. *A distributed tableau for an $\mathcal{ALC}$ distributed TBox $\langle\{\mathcal{T}_s, \mathcal{T}_t\}, \mathcal{B}_{st}\rangle$ is a tuple $D\boldsymbol{T} = \langle\{\boldsymbol{T}_s, \boldsymbol{T}_t\}, \{r_{st}\}_{s \neq t}\rangle$, where $\boldsymbol{T}_s, \boldsymbol{T}_t$ are local $\mathcal{ALC}$ tableaux on local source TBox $\mathcal{T}_s$, and local target TBox $\mathcal{T}_t$ respectively, r_{st} is the mapping relation between TBox $\mathcal{T}_s$ and $\boldsymbol{T}_t$, such that it creates mappings from a subset of axioms in $\mathcal{T}_s$ to a subset of nodes in $\boldsymbol{T}_t$. For an open node $x \in \boldsymbol{T}_t$ and a axiom $A_k \in \mathcal{T}_s$, $(A_k, x) \in r_{ij}$ if there exists at least a complete bridge rule between A_k and x.*

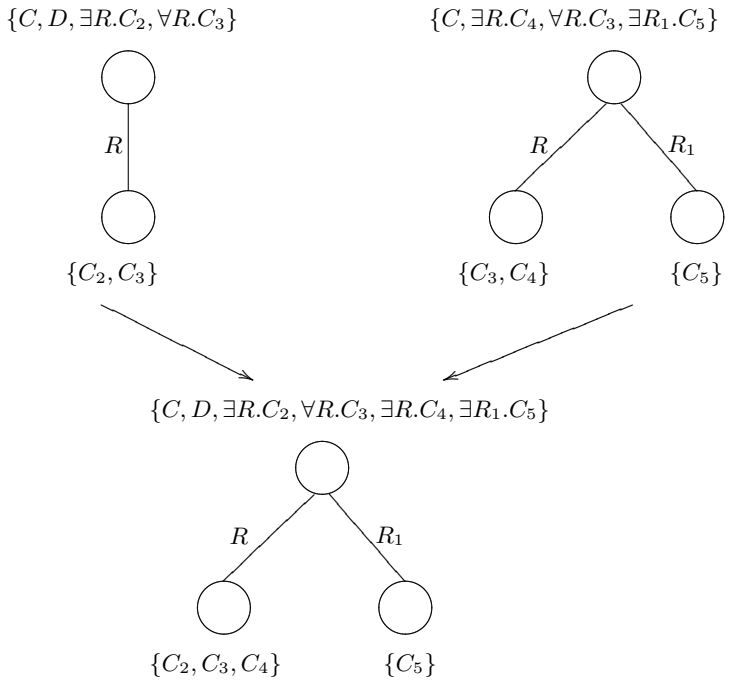

Fig. 1. Merging two nodes of $\mathcal{T}_1$ and $\mathcal{T}_2$

Since in this case, distributed tableaux algorithm implements one set of complete bridge rules, D**T** needs to satisfy the five conditions T1-T5 in section 2.1.4 and the following additional condition:

- T6: If $G \in \mathcal{L}(x), i : G \stackrel{\equiv}{\rightarrow} j : G \in \mathcal{M}_{ij}, G \in Ex(j : A_k)$, and
  $Tab_i(\mathcal{L}(x)) =$ unsatisfiable
  then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{Ex(A_k)\}$

where $\mathcal{M}_{ij}$ is a set of complete bridge rules.

Algorithm

A tableau-based decision procedure for $\mathfrak{T} \models_d C$ is described as follows. Initially we implement the tableau algorithm $\mathbf{T}_1(C)$ and $\mathbf{T}_2(C)$ on two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ simultaneously (in parallel). We will get the two following cases:

1. IF $\mathbf{T}_1(C)$ or $\mathbf{T}_2(C)$ is unsatisfiable (i.e., all the leaf nodes of $\mathbf{T}_1(C)$ or of $\mathbf{T}_2(C)$ are the clashes) then we conclude that C is also unsatisfiable w.r.t $\mathcal{T}$
2. ELSE (both $\mathbf{T}_1(C)$ and $\mathbf{T}_2(C)$ are satisfiable, i.e., there exists at least a clash-free leaf node of $\mathbf{T}_1(C)$ and one of $\mathbf{T}_2(C)$), then we continue applying the tableau algorithm on $\mathcal{T}_1$ for the leaf nodes of $\mathbf{T}_2(C)$ using the identity bridge rules, i.e., $\mathbf{T}_1(\mathbf{T}_2(C))$ is applied only with non-clashes leaf nodes of $\mathbf{T}_2(C)$. Some axioms

of $\mathcal{T}_1$ that contain the concepts and roles in the applied bridge rules between $\mathcal{T}_1$ and $\mathcal{T}_2$ are added to the constraint system for $\mathbf{T}_1(\mathbf{T}_2(C))$.

Input : Concept C
Output : Satisfiable /Unsatisfiable

1: BEGIN
2: $\mathbf{T}_1 = Tab_1(C)$;
3: $\mathbf{T}_2 = Tab_2(C)$;
4: **If** ($\mathbf{T}_1$ is not clashed) and ($\mathbf{T}_2$ is not clashed) **then**
5: $\mathbf{T} = Tab_2(C)$ {execute the local reasoning in $\mathcal{T}_2$ and generate complete tree}
6: **for each** open branch β in $\mathbf{T}$ **do**
7: **repeat**
8: select node $x \in \beta$;
9: $\mathbb{I}_1^{idt}(x) = \{D | 1 : D \xrightarrow{\equiv} 2 : D, D \in \mathcal{L}(x)\}$
10: **if** ($\mathbb{I}_1^{idt}(x) \neq \emptyset$) **then**
11: **repeat**
12: select $D \in \mathbb{I}_1^{idt}(x)$;
13: **if** $D \in Ex(2 : A_k)$ **then**
14: $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{Ex(2 : A_k)\}$
15: **endif**
16: **until** there exist not selected elements in $\mathbb{I}_1^{idt}(x)$
17: **if** ($\mathbf{dTab}_1(\mathcal{L}(x))$) is not satisfiable **then**
18: close {clash in x}
19: break; {verify next branch}
20: **end if**
21: **end if**
22: **until** ((β is open) and (there exist not verified nodes in β))
23: **end for** {all branches are verified}
24: **end If**
25: **if** ($\mathbf{T}$ is clashed) or ($\mathbf{T}_1$ is clashed) or ($\mathbf{T}_2$ is clashed) **then**
26: return unsatisfiable;
27: **else**
28: return satisfiable;
29: **end if**
30: END

Table 8. Distributed tableaux algorithm

Distributed Tableau for Decomposition TBox

To construct a distributed model for a decomposition ontology, we start with a list of initial ABox nodes corresponding to a local ontology. New facts can be added to the ABox forest by applying tableau expansion rules. First of all, nodes will be generated in the target ABox tree $\mathcal{A}_j$ by applying traditional tableau expansion rules. New facts in the open nodes of the target ABox tree should then be sent to the source ABox $\mathcal{A}_i$, i.e., a fact $C(x)$ is generated in an open node $\mathcal{L}(x)$, $\mathcal{L}(x)$ will be sent to the source ABox if there is a bridge rule $i : C \xrightarrow{\equiv} j : C$, therefore $C(x)$ and

$\neg C(x)$ can be generated in the source ABox tree. All clashes can be detected locally. Note that $\mathcal{A}_i$ is built from $\mathcal{L}(x)$ and the axioms contain the bridge rules used. Thus, knowledge propagated in this case is just the axioms, and we can call this process *axiom propagation*. Since the facts are only sent from the target ABox to the source ABox by using the complete bridge rules, there is no message cycle between ABox trees and the termination of the algorithm is still ensured using the blocking strategy.

Theorem 1. *(Termination) For any acyclic DTBox $\mathfrak{T}$ and for any $\mathcal{ALC}$ concept D, distributed tableaux for testing the satisfiability of D terminates.*

Example 6. For a distributed TBox $\mathfrak{T} = \langle\{\mathcal{T}_1, \mathcal{T}_2\}, \mathcal{B}_{12}\rangle$, suppose that $\mathcal{T}_1$ contains an axiom $C \sqcap \exists R.D \sqsubseteq E$, $\mathcal{T}_2$ contains axioms $A \sqsubseteq B$ and $\forall R.B \sqsubseteq C$, and $\mathcal{B}_{12}$ contains two bridge rules $1 : C \xrightarrow{\equiv} 2 : C$ and $1 : R \xrightarrow{\equiv} 2 : R$. A query is proposed to check that concept C subsumes concept description $\forall R.A \sqcap D$ ($\forall R.A \sqcap D$ is subsumed by concept C) w.r.t $\mathfrak{T}$.

This problem can be transformed into checking the satisfiability of $((\forall R.A \sqcap D) \sqcap \neg C)$, see Figure **??**.

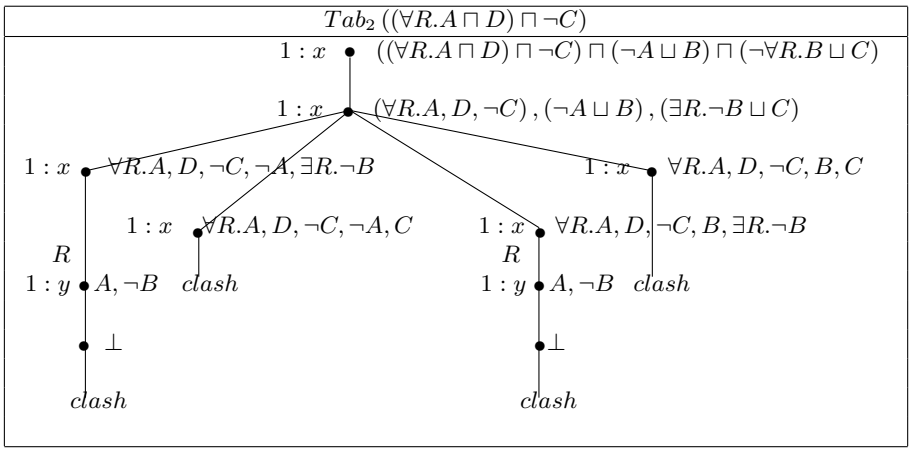

Fig. 2. Example of distributed tableau terminates in $\mathbf{T}_1$

In this example, we stopped at Tab_2 because all the branches in $\mathbf{T}_2$ are closed (all leaf nodes of the local completion tree $\mathbf{T}_2$ are clashed), thus $((\forall R.A \sqcap D) \sqcap \neg C)$ is unsatisfiable w.r.t $\mathcal{T}_2$, i.e., $\mathcal{T}_2 \models \forall R.A \sqcap D \sqsubseteq C$ or $\mathcal{T}_{12} \models_d \forall R.A \sqcap D \sqsubseteq C$. We will address one more example:

Example 7. For a distributed TBox $\mathfrak{T} = \langle\{\mathcal{T}_1, \mathcal{T}_2\}, \mathcal{B}\rangle$, where
$\mathcal{T}_1 = \{AquaticAnimal \sqsubseteq \neg TerrestrialAnimal, TerrestrialAnimal \sqsubseteq Animal\}$,
$\mathcal{T}_2 = \{Fish \sqsubseteq AquaticAnimal\}$,
$\mathcal{B}_{12} = \{1 : AquaticAnimal \xrightarrow{\equiv} 2 : AquaticAnimal\}$.
A query is proposed to check that $Animal$ subsumes $Fish$ ($Fish$ is subsumed by $Animal$) w.r.t $\mathfrak{T}$.

A brief representation for concept and role notations will be convenient to follow the illustration of the algorithm as in Figure **??**, where $\mathcal{T}_1 = \{B_1 \sqsubseteq \neg B_2, B_2 \sqsubseteq A_2\}$,

$\mathcal{T}_2 = \{A_1 \sqsubseteq B_1\}$, and $\mathcal{B}_{12} = \{1 : B_1 \overset{\equiv}{\rightarrow} 2 : B_1\}$. We want to check that A_2 subsumes A_1 (A_1 is subsumed by A_2) w.r.t $\mathfrak{T}$. This problem can be transformed into checking the satisfiability of $A_1 \sqcap \neg A_2$, see Figure **??**.

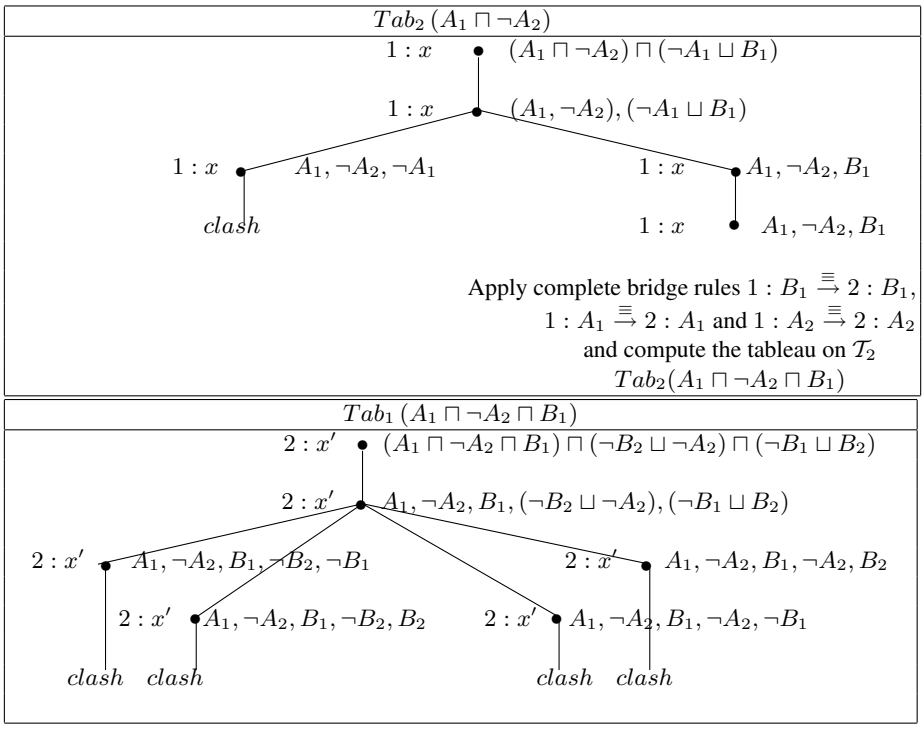

Fig. 3. Example of distributed tableau terminates in $\mathbf{T}_2$

In this example, all open branches of Tab_2 are passed to $\mathcal{T}_1$ by the complete bridge rules and Tab_1 is implemented. All the branches in Tab_1 are then closed, thus $(A_1 \sqcap \neg A_2)$ is unsatisfiable w.r.t $\mathcal{T}_{12}$, in other words $\mathcal{T}_{12} \models_d A_1 \sqsubseteq A_2$.

Theorem 2. *(Soundness and Completeness) An $\mathcal{ALC}$-concept D is satisfiable in distributed TBox $\mathfrak{T}$ iff the distributed tableau construction for D can generate a complete and clash-free completion tree.*

Soundness and completeness of the algorithm follow from the observation that $\mathcal{ALC}$ expansions for decomposition TBox will send concept facts of an open node to the source ABox, and inconsistency is detected when both $C(x)$ and $\neg C(x)$ co-occur in some local ABox. The inconsistency in the distributed ABox must necessarily result in an inconsistency of some local ABox; and an inconsistency in a local ABox will imply that the distributed ABox is also inconsistent.

Theorem 3. *Given a TBox $\mathcal{T}$ and its decomposition presented as a distributed TBox $\mathcal{T}_{12} = \langle \mathcal{T}_1, \mathcal{T}_2, \mathcal{B}_{12} \rangle$, then*

$$\mathcal{T}_{12} \models_d i : X \sqsubseteq j : Y \Leftrightarrow \mathcal{T} \models X \sqsubseteq Y,$$

where X, Y are concepts in $\mathcal{T}$, $i, j \in \{1, 2\}$ and $i \neq j$.

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

Let $\mathcal{I}$ be an interpretation satisfying $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$), with the domain $\Delta^{\mathcal{I}}$, we prove that $\mathcal{I} \models X \sqsubseteq Y$ (or $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$). Indeed,

1. Starting from $\mathcal{I}$, we define $\mathfrak{J} = \langle \{\mathcal{I}_i\}, \{r_{ij}\} \rangle$, where:
   - $\mathcal{I}_i$ is an interpretation with respect to the domain $\Delta^{\mathcal{I}_i}$ and $\Delta^{\mathcal{I}_i} = \Delta^{\mathcal{I}}$. $\mathcal{I}, \mathcal{I}_i$ and $\mathfrak{J}$ are in the same domain (for all $i \in I$). Thus, $\mathcal{I}_i$ interprets every primitive concept, role (or special concept ANYTHING) (i, C) of $\mathcal{T}_i$ by the rule $\#(C)^{\mathcal{I}_i} = C^{\mathcal{I}}$.
   - $\{r_{ij}\}$ are identity relations, i.e., $r_{ij}(C^{\mathcal{I}_i}) \equiv C^{\mathcal{I}_j}$, for all identity bridge-rules between $\mathcal{T}_i$ and $\mathcal{T}_j$.

   First, we need to prove that $\mathcal{I}_i$ and $\mathfrak{J}$ are interpretations of $\mathcal{T}_i$ and $\mathcal{T}_{12}$ respectively. We can easily show that $(\Delta_i \, , \cdot^{\mathcal{I}_i})$ is an interpretation of $\mathcal{T}_i$, because:

$$\Delta^{\mathcal{I}_i} \text{ is not empty} \tag{1}$$
$$ANYTHING^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i} \tag{2}$$
$$NOTHING^{\mathcal{I}_i} = \emptyset \tag{3}$$
$$M^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i} \text{ for every concept } M \text{ of } \mathcal{T}_i \tag{4}$$
$$R^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i} \text{ for all the roles } R \text{ of } \mathcal{T}_i \tag{5}$$

   According to the overlap decomposition, for each defined concept (role) M of $\mathcal{T}$, there exists a defined concept (role) $\#M$ in $\mathcal{T}_i (i \in I)$ and $\#(M)^{\mathcal{I}_i} = M^{\mathcal{I}}$ (because $\mathcal{I}_i$ and $\mathcal{I}$ are in the same domain).
   Moreover, we have $\#(M)^{\mathcal{I}_i} = M^{\mathcal{I}}$, for every arbitrary concept (role) M in $\mathcal{T}_i$ (follows from Proposition 1 in Subsection 3.3, $\#(M)^{\mathcal{I}_i} = (i, M)^{\mathcal{I}_i} = M^{\mathcal{I}}$). Consequently,

$$\#(M)^{\mathcal{I}_i} \subseteq \#(N)^{\mathcal{I}_j} \Leftrightarrow M^{\mathcal{I}} \subseteq N^{\mathcal{I}}$$

   For each axiom $(i, V) \sqsubseteq (i, W)$ of $\mathcal{T}_i$, by the overlap decomposition, there exists an axiom $V \sqsubseteq W$ of $\mathcal{T}$.
   Since $\mathcal{I}$ satisfies $\mathcal{T}$ and $(V \sqsubseteq W) \in \mathcal{T}$, consequently $\mathcal{I} \models V \sqsubseteq W \Rightarrow V^{\mathcal{I}} \subseteq W^{\mathcal{I}}$. In addition, $\#(M)^{\mathcal{I}_i} = \#(N)^{\mathcal{I}_i} \Leftrightarrow M^{\mathcal{I}} = N^{\mathcal{I}}$, therefore

$$\begin{aligned} V^{\mathcal{I}} \subseteq W^{\mathcal{I}} \Rightarrow \#(V)^{\mathcal{I}_i} \subseteq \#(W)^{\mathcal{I}_i} \Leftrightarrow \mathcal{I}_i \models (i, V) \sqsubseteq (i, W) \\ \Rightarrow \mathfrak{J} \models_d i : V \sqsubseteq W \end{aligned} \tag{6}$$

   From (1)–(6) above, we obtain

$$\mathcal{I}_i \models \mathcal{T}_i \Rightarrow \mathfrak{J} \models_d \mathcal{T}_i \tag{7}$$

   For each identity bridge-rule $i : V \underset{\rightrightarrows}{\equiv} j : V$. Since $V^{\mathcal{I}} \equiv \#(V)^{\mathcal{I}_i}$ and $V^{\mathcal{I}} \equiv \#(V)^{\mathcal{I}_j} \Rightarrow \#(V)^{\mathcal{I}_i} \equiv \#(V)^{\mathcal{I}_j}$

$$\Rightarrow \mathfrak{J} \models i : V \underset{\rightrightarrows}{\equiv} j : V \tag{8}$$

   From (7) and (8), we have $\mathfrak{J} \models \mathcal{T}_{12}$.

2. Now, by the hypothesis and $M^{\mathcal{I}} = \#(M)^{\mathcal{I}_i}$ we deduce that

$$\begin{aligned}\mathfrak{J} \models i : X \sqsubseteq j : Y &\Rightarrow \#(X)^{\mathcal{I}_i} \subseteq \#(Y)^{\mathcal{I}_j} \\ &\Rightarrow X^{\mathcal{I}} \subseteq Y^{\mathcal{I}} \\ &\Rightarrow \mathcal{I} \models X \sqsubseteq Y\end{aligned}$$

($\Longleftarrow$) $\mathcal{T} \models X \sqsubseteq Y \Rightarrow \mathcal{T}_{12} \models i : X \sqsubseteq j : Y$
Let $\mathfrak{J} = \langle \{\mathcal{I}_i\}, \{r_{ij}\} \rangle$ be a d-interpretation satisfying $\mathcal{T}_{12}$, where $\mathcal{I}_i$ is an interpretation of $\mathcal{T}_i$, $\{\mathcal{T}_i\}_{i \in I}$ are in the same domain, $\{r_{ij}\}_{i \neq j}$ are identity relations between two domains $\Delta^{\mathcal{I}_i}$ and $\Delta^{\mathcal{I}_j}$. It should be shown that

$$\mathfrak{J} \models i : X \sqsubseteq j : Y((X)^{\mathcal{I}_i} \subseteq \#(Y)^{\mathcal{I}_j})$$

We define an interpretation $\mathcal{I}$, with the domain $\Delta \equiv \Delta^{\mathcal{I}_i}, \forall i \in I$, that interprets a primitive concept (role) C of $\mathcal{T}$ using the rule $C^{\mathcal{I}} = \#(C)^{\mathcal{I}_i}$.
We need to prove that $\mathcal{I} \models \mathcal{T}$. Indeed, for each arbitrary concept (role) M of $\mathcal{T}$, there exists a concept (role) correspondent $\#M$ in $\mathcal{T}_i$ and we have

$$M^{\mathcal{I}} = \#(M)^{\mathcal{I}_i} \text{(follows the above proof)} \tag{9}$$

For each axiom $V \sqsubseteq W$ of $\mathcal{T}$, by the overlap decomposition, there exists an equivalent axiom $\#V \sqsubseteq \#W$ in $\mathcal{T}_i$.
Because $\mathfrak{J} \models_d T_{12} \Rightarrow \mathfrak{J} \models_d T_i, \forall i \in I \Rightarrow \mathcal{I}_i \models \mathcal{T}_i$
Starting from $\#V \sqsubseteq \#W \in \mathcal{T}_i$, we have $\mathcal{I}_i \models \#V \sqsubseteq \#W \Rightarrow \#(V)^{\mathcal{I}_i} \subseteq \#(W)^{\mathcal{I}_i} \Rightarrow V^{\mathcal{I}} \subseteq W^{\mathcal{I}}$

$$\Rightarrow \mathcal{I} \models V \sqsubseteq W \tag{10}$$

From (9) and (10), $\Rightarrow \mathcal{I} \models \mathcal{T}$.
Otherwise, from $\mathcal{T} \models X \sqsubseteq Y$ (by the hypothesis) $\Rightarrow \mathcal{I} \models X \sqsubseteq Y \Rightarrow X^{\mathcal{I}} \subseteq Y^{\mathcal{I}} \Rightarrow \#(X)^{\mathcal{I}_i} \subseteq \#(Y)^{\mathcal{I}_j} \Rightarrow \mathfrak{J} \models i : X \sqsubseteq j : Y$

The main message of Theorem 1 is that in decomposition TBox, the decision whether $\mathcal{T}_{12} \models_d i : X \sqsubseteq j : Y$ corresponds to a standard subsumption relation in $\mathcal{T}$, $\mathcal{T} \models X \sqsubseteq Y$.

4.4 Discussion

This section will discuss the distinction between ontology decomposition and distributed description logic that conducted reasoning to parallel and distributed techniques as mentioned above.

Ontology Decomposition and Distributed Description Logic

DDL is an extension of DL starting from one of the challenges in the semantic web, that of being able to solve the reasoning problem with a large number of overlapping and heterogeneous local ontologies, in which each ontology describes an application domain from a local and subjective perspective. For example, to describe people, ontology 1 uses the concept "PERSON" whereas ontology 2 uses "COUPLE". There are

clear relations between these two ontologies, e.g., if two persons in ontology 1 have a marriage relationship, then they make a couple in ontology 2. These interrelationships are represented by semantic mappings (bridge rules). Local ontologies, however, are independent, autonomous, and interpret different domains. Hence, reasoning and query handling are only implemented locally, i.e., a query is handled on one local ontology and knowledge can be propagated from other local ontologies through semantic mappings. Reasoning in DDL is based essentially on a global ontology that is combined from all local ontologies and semantic mappings between them [**?**, **?**]. In fact, this approach meets some drawbacks as mentioned in [**?**]. The first one is that the cost in space for reasoning with the global ontology is much larger than the sum of space costs for reasoning with the local ontologies. Thus a suitable combination of local ontologies is necessary for global reasoning. The second one is that the global reasoning loses the autonomy of local ontologies, i.e., reasoning in the local ontologies can be done by specific reasoners that are optimized by local languages, whereas reasoning in the global ontology has to be performed by the most general reasoner which is capable of dealing with the most general local language. The third one is that in some cases the combination of local ontologies as a whole is not available, this can lead to an inconsistency in the global ontology or limit the access to the local ontologies.

Our approach, ontology decomposition, starts from the opposite idea where we want to divide a large ontology into multiple smaller ontologies and to reason in the systems of these ontologies. We have chosen to work with ontology decomposition because it provides the highest degree of de-coupling between different ontologies. This is important for our purposes as we want to support localized reasoning. Local ontologies in this case are in the same domain as the original ontology since they share the set of concepts and roles of the original ontology. Ontology decomposition can be examined as a particular case of DDL, where its semantic mappings are only identity relations between concepts (roles) that co-occur in two axioms of different ontologies. The most important results are the preservation of semantic and inference services between the original ontology and the decomposition ontology. Furthermore, queries can be handled simultaneously and return the results on local ontologies. A query can be also decomposed into sub-queries which are then handled separately on the local ontologies, and the results are then combined. When a query is given, a set of bridge rules (extracted from the concepts and roles that occur in the query expression) will be added into the complete bridge rules. This is to support propagating the knowledge from the other ontologies, i.e., in fact, there are may be some concepts (roles) in the query expression that were not present in the bridge rules, and the reasoning will not be adequate if we lack the knowledge that needs to be transfered from these added bridge rules. Thus the reasoning in source TBoxes is only implemented with the axioms that contribute to the presence of complete bridge rules.

From characteristics of the ontology decomposition, we have proposed some reasoning techniques as mentioned in the above section.

Parallel and distributed reasoning

In parallel reasoning, local tableaux are created simultaneously on local ontologies, open nodes in local tableaux are then merged and combined for the results. Merging is done through the set of complete bridge rules. For larger ontologies which may be

difficult to decompose into disjointed partitions, this method is very effective. However, the cost for merging is large if there are a large number of open nodes in the local ontologies. Queries can be also decomposed into sub-queries and are settled in parallel on local ontologies, with query answers subsequently combined.

In distributed reasoning, we minimize the number of axioms (GCIs) used. The results are obtained on local ontologies. This approach is suitable for larger ontologies that can be partitioned into disjoint sub-ontologies as the propagating rules are applied as little as possible.

However, for ontologies where the decomposed ontologies are complete disconnected (the set of bridge rules is empty), reasoning results can be concluded directly on the local ontologies without the need for merging (as in the parallel case) nor for axiom propagation (as in the distributed case).

5 Conclusion and future work

In this paper, we have proposed two algorithms for reasoning in a decomposed ontology based on parallel and distributed approaches. Previous optimizations for reasoning are only effective in certain knowledge bases and some special structures. Decomposing an ontology into smaller ontologies contributes to improving the speed of reasoners, which operate on the sub-ontologies, while preserving the reasoning results of the original ontology. This technique is applicable to most real-life ontologies, especially for ontologies with large numbers of GCI axioms and a complex structure. The best results of decomposition preserve the reasoning services of the original ontology. In cases where there may be little beneficial effect, at least the technique does not falsify results.

Although concrete decomposition techniques were not presented in this paper, some essential properties of decomposition that influence the reasoning performance have been provided. The effective optimizing algorithms for decomposing a large ontology will be introduced in a subsequent paper. This again depends on having an effective (and cheap) method for analyzing the likely characteristics of a given test ontology. We are also performing more experiments with very large KBs, e.g., UMLS, for showing the effect of reasoning over decomposed ontologies. Reasoning in multiple decomposed ontologies (the case where an ontology is decomposed into many sub-ontologies) by combining both parallel and distributed methods will be the subject of a future paper. We are embarking on optimizing the decomposition algorithm using graph theory and treating queries effectively in DDL. An efficient solution for decomposition into several sub-ontologies and reasoning over those sub-ontologies will be proposed.

Acknowledgements

A preliminary version of this paper has been published in *the CE2006 international conference* [**?**], and we would like to thank all the reviewers for their very useful comments.

References

[1] F. 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 on Principles of Knowledge Representation and Reasoning, KR-92, pages 270-281, Boston (USA), 1992*

[2] F. Baader and W. Nutt, Basic Description Logics. *Spinger, 2003*.

[3] J. Bao, D. Caragea and V. Honavar, A Distributed Tableau Algorithm for Package-based Description Logics. *In Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence, pages 404-410. 2006*.

[4] J. Bao, D. Caragea and V. Honavar, On the Semantics of Linking and Importing in Modular Ontologies. *In I. Cruz et al. (Eds.): ISWC 2006, LNCS 4273, pages 72-86. 2006*.

[5] A. Borgida, L. Serafini, Distributed Description Logics: Assimilating information from peer sources. *Journal of Data Semantics, 1:153-184, 2003*.

[6] F. M. Donini, Complexity of Reasoning. *Description Logic Handbook 2003: 96-136*.

[7] J. W. Freeman, Improvements to Propositional Satisfiability Search Algorithms. *Dissertation in Computer and Information Science, 1995*.

[8] V. Haarslev and R. Möller, High Performance Reasoning with Very Large Knowledge Bases. *DL-2000*.

[9] V. Haarslev and R. Möller, Optimizing TBox and ABox Reasoning with Pseudo Models. *DL-2000*.

[10] I. Horrocks, Optimizing Tableaux Decision Procedures for Description Logics. *PhD thesis, University of Manchester, 1995*.

[11] I. Horrocks and U. Sattler, Optimised Reasoning for $\mathcal{SHIQ}$. *In Proc. of the 2005 Description Logic Workshop (DL 2005), volume 147 of CEUR, 2005*.

[12] I. Horrocks and P. F. Patel-Schneider, Fact and dlp. *In Proc. of the Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 1998), pages 27-30, 1998.*

[13] B. Nebel, Terminological Reasoning is Inherently Intractable. *Artificial Intelligence, 43:235-249*.

[14] T. A. L. Pham, Raisonnement sur des ontologies décomposées. *Rapport de recherche, Lab I3S, Université de Nice-Sophia Antipolis, 2006*.

[15] T. A. L. Pham, N. Le-Thanh, Decomposition-based Reasoning for Large Knowledge Bases in Description Logics. *Proceedings of the 13th ISPE International Conference on Concurrent Engineering: Research and Applications, Antibes, French Riviera, France, 18-22 September 2006*.

[16] T. M. Pham, 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*.

[17] L. Serafini and A. Tamilin, DRAGO: Distributed Reasoning Architecture for the Semantic Web. *Technical Report T04-12-05, ITC-irst, December 2004*.

[18] E. Sirin et al., Optimizing Description Logic Reasoning with Nominals: First Results. *UMIACS Technical Report 2005-64*.