HIGH PERFORMANCE ABSORPTION ALGORITHMS FOR TERMINOLOGICAL REASONING IN DESCRIPTION LOGICS

Ming Zuo

A thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

September 2006

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Ming Zuo**

Entitled: **High Performance Absorption Algorithms for Terminological Reasoning in Description Logics**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair
Dr. Hon Fung Li

_____ Examiner
Dr. Greg Butler

_____ Examiner
Dr. Nematollaah Shiri

_____ Supervisor
Dr. Volker Haarslev

Approved _____
Chair of Department or Graduate Program Director

_____ 20 _____ _____
Dr. Nabil Esmail, Dean
Faculty of Engineering and Computer Science

Abstract

High Performance Absorption Algorithms for Terminological Reasoning in Description Logics

Ming Zuo

When reasoning with description logic (DL) knowledge bases (KBs) which contain a large number of axioms, performance is the key concern in real applications. To improve the performance, axiom absorption has been a central research issue in DL KBs. Well-known algorithms for axiom absorption, however, still heavily depend on the order and the format of the axioms occurring in the target KB. In addition, in many cases, there exist some restrictions in these algorithms which prevent axioms from being absorbed. Both the characteristics and the design of absorption algorithms for optimal reasoning are still open problems.

In this thesis, we first seek to improve our theoretical understanding about the axiom absorption techniques including some related techniques such as simplification and normalization. Then we propose a criterion for the "best" absorption against experimental experience. Based on this criterion, we develop some new algorithms to absorb axioms in a KB to ameliorate the reasoning performance.

The experimental tests we conducted are mostly based on synthetic benchmarks derived from common cases will occur in real KBs. The experimental evaluation demonstrates a significant runtime improvement.

Acknowledgments

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank the Department of Computer Science of Concordia University for giving me permission to commence this thesis in the first instance, to do the necessary research work and to use departmental data. I am deeply indebted to my supervisor Prof. Dr. Volker Haarslev whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis.

My colleagues from the Department of Computer Science supported me in my research work. I want to thank them for all their help, support, interest and valuable hints. Especially I am obliged to Yu Ding, Hsueh-Ieng Pai, Xi Deng, Jiaoyue Wang, Cuiming Chen and Weisheng Lin. My friend Lianyun Chu looked closely at the final version of the thesis for English style and grammar, correcting both and offering suggestions for improvement.

I owe my loving thanks to my wife Kelly Wang, my daughter Yufei Zuo. They have lost a lot due to my research. Without their encouragement and understanding it would have been impossible for me to finish this work. My special gratitude is due to my brothers, my sisters and their families for their loving support.

Contents

List of Figures

Chapter 1

Introduction

When reasoning with description logic (DL) knowledge bases (KBs) which contain a large number of axioms, performance is the key concern in real applications. To improve the reasoning performance, many optimization algorithms and techniques are employed by most modern reasoners such as RACER, FaCT++, and Pellet. Among the optimization algorithms, lazy unfolding is proven to be one of the most effective [8]. However, lazy unfolding does not work well for KBs containing a significant number of unabsorbable General Concept Inclusions (GCIs). A GCI is called *unabsorbable* if it can not be rewritten into a *rule axiom*. We use the term *rule axiom* to represent the axioms of the form $(A \Rightarrow C)$ where $A \in NC$[1] and C is an arbitrary concept, while the form $(C \sqsubseteq D)$ represents a GCI where C and D are arbitrary concepts. The difference between $(A \Rightarrow C)$ and $(A \sqsubseteq C)$ is that $(A \Rightarrow C)$ only represents one-way reasoning "if A then C", while $(A \sqsubseteq C)$ represents both "if A then C" and "if $\neg C$ then $\neg A$". To convert a GCI axiom into a rule axiom, a technique called *absorption* is employed. Although preliminary absorption algorithms have been introduced and discussed by Horrocks & Tobies [7] and Haarslev & Möller [5], the result of these algorithms tightly depends on the axiom order and axiom format of the input KB. There was little concern about how to find the "best" absorption among

[1] In this thesis, we use NC to represent the set of concept names.

many absorption choices. In addition, some restrictions in these algorithms prevent many axioms from being absorbed. The demand of absorption algorithms for optimal reasoning becomes critical when the KB based systems with large number of axioms are widely developed in the fields of the Semantic Web, medical, and bio-information applications.

In this thesis, we first study the our theoretical issues about absorption and its related optimization techniques such as normalization and simplification. Then we define a criterion for "best" absorption. Based on this criterion, we develop some new absorption formula and absorption algorithms by extending the current well-known algorithms. To evaluate the effectiveness of these newly developed absorption techniques, we developed a prototype system and conducted some experiments. The experimental tests we mostly used synthetic benchmarks derived from common cases found in real KBs. At last, we identify several interesting research directions for future work.

Before we discuss the absorption algorithms in more detail, we first present an introduction to description logics and other preliminaries of our work.

1.1 Preliminaries

1.1.1 Knowledge Bases and Description Logics

Compared to a database system, one of the significant characteristics of a knowledge-based system is that it has the ability to find implicit knowledge from its explicitly represented knowledge. This ability in computer science is often called *reasoning*. Consequently, there are two main topics for knowledge-based system research — *knowledge representation*, i.e., the approaches to represent knowledge, and *reasoning*, i.e., the approaches to derive implicit knowledge.

The approaches of representing knowledge have been developed since the 1970's,

and are sometimes divided into two categories: *logic based* formalisms and *non-logic based* representations. First-order logic is the typical example of a logic-based representation. For the latter, semantic networks and frames are specialized representations. Accordingly, we use the term *network-based representation structures* to refer to non-logic based approaches [1].

Logic-based approaches are more generally applicable from the very start, and they provide a powerful and general machinery. Since first-order logic or other logic-based representations are introduced by most computer science introductory text-books, in the following discussion, we will mainly concentrate on introducing *network-based representation structures* and its combination with logic-based knowledge representations. After that, we will introduce the origin and basic features of description logics. In the meantime, we will also introduce some basic terminologies which are widely used in DL and KB systems.

In *network-based representation structures*, the basic elements are *nodes* and *links*. *Nodes* are used to represent *concepts*. In KB systems, a *concept* represents a set of individual objects. In accordance with that, an individual of a concept is called an *instance* of that concept. *Links* are used to represent relationships between *concepts*. In addition, a *concept* have *properties* — also called *attributes* in some systems — which are simply attached to the corresponding *nodes*. There is a typical example of *network-based representation structures* as shown in Figure 1 [11]. The entire structure is also referred to as a *terminology* which indeed represents the generality of the concepts involved. Between concepts, the link ($\Longrightarrow$) is called *IS-A* relationship. For example, the relationship between **Mother** and **Female** means that *mothers* are *females*. An *IS-A* relationship defines a hierarchy over the concepts and provides the basis for the *inheritance of properties* — i.e., when a concept is more specific than some other concepts, it inherits the properties of the more general one. For example, if a *parent* has a *hasChild* property and the number of its children is (≥ 1), then

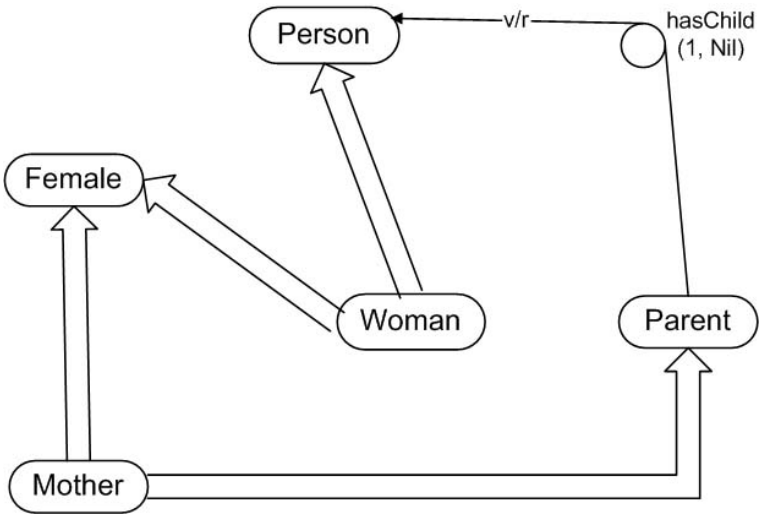

Figure 1: An example of network-based representation structures

a *mother* should also have the *hasChild* property and the number of her children is also (≥ 1), because **Mother** is more specific than **Parent**. A *property* of a concept is also called *role*. A *role* might have what is called *value restriction*, denoted by the label v/r in Figure 1 which expresses a limitation on the range of objects that can fill that *role*. The value 1 in number restriction (1, NIL) denotes the lower bound on the number of children, and the second value denotes the upper bound, and NIL denotes infinity. In this case, the representation of **Parent** can be read as *a* parent *is a* person *who has at least one child, and everyone of his/her children is a* ***Person***.

A characteristic feature of *network-based representation structures* is the ability to infer implicit relationships. For example, from Figure 1, if we define **Woman** as the concept of **Female Person**, then we can infer that a *mother* is also a *woman* because **Mother** is both a **Female** and a **Person**.

From a practical point of view, *network-based representation structures* were often considered more appealing and more effective than logic-based systems because of their human-centered origins. However, the more complicated the relationships established among concepts, the more difficult it becomes to give a precise characterization of the kind of relationships can be computed, and how this can be done

correctly without failing to recognize some of the relationships.

In fact, *network-based representation structures* are not fully satisfactory because of their lack of precise semantics [1]. In many cases, for even virtually identically looking components, one system might behave very differently from others. As a result, the need emerged for providing semantics to representation structures and gaining both simple representation and efficient reasoning.

The semantics of *network-based representation structures* can be given by defining a language for the elements of the structure and by providing an interpretation for the strings that language. This motivation ultimately led to the development of description logics and description languages.

1.1.2 The Description Language

The basic step to construct this language is to provide two disjoint alphabets of symbols that are used to denote *atomic concepts* and *atomic roles*. *Atomic concepts* are unary predicate symbols, denoting sets of individuals; and *atomic roles* are binary predicate symbols, used to express relationship between concepts. To distinguish the function of each concept in the role relationship, the concept that corresponds to the second argument of the role is called *role filler*. For example, in Figure 1, **Person**, **Mother**, **Parent**, **Female**, and **Woman** are *atomic concepts*, and *hasChild* is an *atomic role*.

Complex descriptions can be built inductively by using set operators such as *intersection*, *union*, and *complement of concepts*. Thus, we can describe the concept of ***Person*** *who is not a* ***Woman*** as $\mathbf{Person} \sqcap \neg \mathbf{Woman}$, and the concept of *individuals who are* ***Male*** *or* ***Female*** as $\mathbf{Female} \sqcup \mathbf{Male}$.

To emphasize the relationship to logic, *intersection*, *union*, and *complement of concepts* are also referred to as *concept conjunction*, *concept disjunction*, and *concept negation*, respectively.

Now let us turn our attention to the expression of role restrictions in Figure 1. Most description languages provide (full) existential qualifications and value restrictions for role restrictions. Based on these restrictions, we can describe the kind of concept, such as, *"persons having child"* as $\exists hasChild.\mathbf{Person}$ and *"individuals all of whose children are* ***Persons****"* as $\forall hasChild.\mathbf{Person}$. Therefore, in addition to Figure 1 in which the concept **Parent** is expressed by using *nodes* and *links*, we can also describe this concept by using logic expressions. This is shown in Figure 2.

(i)$\mathbf{Parent} \equiv \mathbf{Person} \sqcap \exists hasChild.\mathbf{Person}$
(ii)$\mathbf{Mother} \equiv \mathbf{Parent} \sqcap \mathbf{Female}$
(iii)$\mathbf{Woman} \equiv \mathbf{Female} \sqcap \mathbf{Person}$

Figure 2: A DL terminology example

We call this kind of language *Description Language*, and the underlying logics *Description Logics*. In description languages, the basic elements are *atomic concepts* and *atomic roles*. Complex descriptions can be built from them inductively by using *concept constructors*. To simplify our discussion, in the following part we will use the letters A and B for *atomic concepts*, the letters R and S for *atomic roles*, and the letters C and D for general concept descriptions if there is no other specification.

Description languages are distinguished by the *concept constructors* they provide. The most common description language is called $\mathcal{AL}$ (attribute language). Most of other description languages can be seen as extensions of the $\mathcal{AL}$ language.

In $\mathcal{AL}$, a concept is formed according to the following constructors [1]:

A	(atomic concept)
$\top$	(universal/top concept)
$\bot$	(bottom concept)
$\neg$A	(atomic negation)
C$\sqcap$D	(intersection)
$\forall R.C$	(value restriction)
$\exists R.\top$	(limited existential quantification)

Note that in $\mathcal{AL}$, negation can only be applied to *atomic concepts*, and the only concept allowed in the role filler of existential quantifications is the top concept $\top$.

More expressive languages can be obtained if we add more constructors to $\mathcal{AL}$. If the union connective is also used to construct concept such as $(C \sqcup D)$, the $\mathcal{AL}$ extension language is called $\mathcal{ALU}$. If the full existential quantification is used, such as $(\exists R.C)$, then the $\mathcal{AL}$ extension language is called $\mathcal{ALE}$. Number restrictions are written as $(\geq nR)$ (at-least restriction) and $(\leq nR)$ (at-most restriction), where n is a nonnegative integer. This extension is called the $\mathcal{ALN}$ language. The negation of an arbitrary concept written as $(\neg C)$, is called $\mathcal{ALC}$ ($\mathcal{C}$ for compliment). The full extension of the $\mathcal{AL}$ language is called $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}]$. In the following discussion, whenever we mention description language, we refer to the full extension of the $\mathcal{AL}$ language. For ease of expression, we will use $\mathcal{ALCN}$ instead of $\mathcal{ALUENC}$ to represent this description language.

1.1.3 Reasoning with Description Logics

Within a KB, the general knowledge of the problem domain is called *intentional knowledge* such as the above mentioned terminology; the knowledge to the particular problem is called *extensional knowledge*. For example, the restriction of *"***Parents** *are individuals having child"* is *intentional knowledge*. In contrast, the fact that *"Mary is a* **Mother***"* is *extensional knowledge*. For a system using DL language

in its KB, we also call the two parts as TBox and ABox respectively. The TBox contains intentional knowledge in the form of a terminology (hence the term TBox, abbreviated as $\mathcal{T}$ in the following discussions) which is built through declarations that describe general properties of concepts. The ABox contains extensional knowledge — also called assertional knowledge (hence the term ABox, abbreviated as $\mathcal{A}$ in the following discussions) — knowledge that is specific to the domain of interest.

As we have already mentioned, reasoning is the basic feature of a knowledge-based system. A knowledge representation system based on DL must have the capability to perform reasoning, i.e., a knowledge representation system goes beyond expressing concept definitions and assertions — it contains implicit knowledge that can be made explicit through reasoning (*inferences*).

Before continuing our discussion, we give an example to demonstrate how *inference* works for a DL KB system. Consider the example TBox $\mathcal{T}$ described in Figure 2, and an ABox described as follows:

Mary:hasChild(Tom);
Mary:**Female**;
Mary:**Person**;
Tom:**Person**

Suppose we need to answer the question: *Is Mary a* **Mother**? Obviously, based on both the TBox and ABox above, we are unable to answer this question by only searching the explicit knowledge as we do in a database system (In a database system, our answer is *false* if we fail to find an explicit answer directly). In a KB system, however, the answer might not be correct if we do so. Please refer to Section 1.1.4. In a knowledge-based system, we have to answer such kind of questions by *inference*. First, suppose *Mary* is not a **Mother**, expressed as:

(1)$Mary : \neg$**Mother**

By considering axiom (ii) in $\mathcal{T}$, we have the following assertion by substituting **Mother** with its definition:

$$(2) Mary : \neg(\textbf{Parent} \sqcap \textbf{Female})$$

The above expression can be rewritten as the following by moving negation in front of concept names.

$$(3) Mary : \neg\textbf{Parent} \sqcup \neg\textbf{Female}$$

Now, the above expression has two parts connected by a disjunction, and its semantics can be interpreted as "either *Mary* is not a **Parent** or *Mary* is not a **Female**". "*Mary* is not a **Female**" conflicts with the statement "Mary is a **Female**" in the ABox. Now let us consider the first part: "*Mary* is not a **Parent**". Consider axiom (i) in $\mathcal{T}$, we have the following assertion by replacing **Parent** with its definition and moving negation only in front of concept names:

$$(4)\ Mary : \neg\textbf{Person} \sqcup \forall\textbf{hasChild}.\neg\textbf{Person}$$

Now the assertion is comprised of two parts: the first part that states "*Mary* is not a **Person**" conflicts with "*Mary* is a **Person**" in the ABox; the second part that states "all of *Mary's* children are not **Persons**" conflicts with "*Tom* is a child of *Mary* and *Tom* is a **Person**" in the ABox. Obviously, the assumption that "*Mary* is not a **Mother**" does not hold. Thus, we infer that "*Mary* is a **Mother**" holds.

The kind of logical inference we just discussed is called *satisfiability* in which we tried to test if an individual *Mary* can be an instance of the concept $\neg$**Mother**. The procedure we used to expand a concept according to a TBox is called *unfolding*. Other logical inferences frequently used in a DL based KB system also include *subsumption*, *equivalence*, and *disjointness*. We will discuss these inferences later in detail.

1.1.4 Closed-world versus Open-world Assumptions

It is useful to compare database systems with knowledge-base systems. The analogy is that the schema of a database is compared to the TBox and the instance with the actual data is compared to the ABox. However, the semantics of ABoxes differs from the usual semantics of database instances. While a database instance represents exactly one interpretation, namely the one where classes and relations in the schema are interpreted by the objects and tuples in the instance, an ABox represents many different interpretations, namely all its models. As a consequence, absence of information in a database instance is interpreted as negative information, while absence of information in an ABox only indicates lack of knowledge.

For example, if the only assertion about Peter is *Peter:hasChild(Harry)*, then in a database this is understood as a representation of the fact that Peter has only one child, Harry. In an ABox, the assertion only expresses that, in fact, Harry is a child of Peter. However, the ABox has several models, some in which Harry is the only child and others in which Harry has siblings. Consequently, even if one also knows that Harry is male, one cannot deduce that all of Peters children are males. The only way of stating in an ABox that Harry is the only child is by doing so explicitly, that is by adding the assertion $Peter{:}(\leq 1)hasChild.\mathbf{Person}$. This means that, while the information in a database is always understood to be complete (the assumption of closed-world), the information in an ABox is in general viewed as being incomplete (the assumption of open-world). The semantics of ABoxes is therefore sometimes characterized as an "open-world" semantics, while the traditional semantics of databases is characterized as "closed-world".

This view has consequences for the way queries are answered. From a logical point of view, querying in database systems means a finite look up. Since an ABox represents possibly infinitely many interpretations, namely its models, query answering is more complex: it requires nontrivial reasoning. Thus, querying in knowledge-based

systems can be very complex.

To illustrate the difference between closed-world versus open-world semantics, we will discuss the famous so-called Oedipus example, which has stimulated a number of theoretical developments in DL research [1].

Example 1.1 According to the Oedipus story from ancient Greek mythology, Oedipus killed his father; married his mother Iokaste; and had children with her, among them Polyneikes. Also, Polyneikes had children, among them Thersandros.

To represent the rudimentary facts, we build the following ABox $\mathcal{A}_{oe}$ in Figure 3. To simplify our discussion, we use an atomic concept **Patricide** to represent an individual who murdered his or her own father.

hasChild(IOKASTE, OEDIPUS)	hasChild(IOKASTE, POLYNEIKES)
hasChild(OEDIPUS, POLYNEIKES)	hasChild(POLYNEIKES, THERSANDROS)
Patricide(OEDIPUS)	¬Patricide(THERSANDROS)

Figure 3: The Oedipus ABox $\mathcal{A}_{oe}$

Suppose we now want to know from the ABox whether IOKASTE has a child that is a **Patricide** and that child itself has a child who is not a **Patricide**. This can be expressed as follows:

$$\mathcal{A}_{oe} \models (\exists hasChild.(\mathbf{Patricide} \sqcap \exists hasChild.\neg\mathbf{Patricide}))(IOKASTE)$$

The symbol "$\models$" denotes the entailment which means the right-hand side is a logical consequence of the left-hand side.

By applying the closed-world assumption, we may do the following reasoning: IOKASTE has two children: OEDIPUS and POLYNEIKES. Nothing tells us that POLYNEIKES is a **Patricide**, so POLYNEIKES is not the child we are looking for. OEDIPUS is a **Patricide**, and he has a child POLYNEIKES. Again, nothing tells us that POLYNEIKES is not a **Patricide**. Thus OEDIPUS is not the child we are looking for. Therefore, we give the answer that *IOKASTE does not have this kind of child*.

However, by applying the open-world assumption, we may do reasoning as the following: POLYNEIKES is a child of IOKASTE, and POLYNEIKES can only be either **Patricide** or not a **Patricide**. Suppose POLYNEIKES is a **Patricide**, since POLYNEIKES has a child THERSANDROS who is not a **Patricide**. Then POLYNEIKES is the child we are looking for, and the answer is *yes*. Then suppose that POLYNEIKES is not a **Patricide**. Because IOKASTE has a child OEDIPUS who is a **Patricide** and OEDIPUS has child POLYNEIKES who is not a **Patricide**, OEDIPUS is the child we are looking for. Again, the answer is *yes*.

As the above example shows, open-world assumption reasoning may require case analysis. Therefore, inferences in knowledge-based systems are obviously more complex than query processing in databases.

1.1.5 Rules

In general, the knowledge bases we considered so far consist of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. We use $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ to denote a knowledge base.

In some DL systems, *rules* are also used to express knowledge as well as the terminologies we discussed above in a TBox $\mathcal{T}$. The simplest variant of such *rules* are expressions in the form of $(C \Rightarrow D)$ where C and D are concepts. The meaning of this rule is "if an individual is proven to be an instance of C, then it is also an instance of D." Such rules are also called *triggers* which are used active DB systems.

Note that *rule* $(C \Rightarrow D)$ is not the same as the terminology $(C \sqsubseteq D)$. $(C \Rightarrow D)$ emphasizes only one-way reasoning "if C then D" by definition, while the terminology $(C \sqsubseteq D)$ represents two way reasoning, i.e., both $(C \Rightarrow D)$ and $(\neg D \Rightarrow \neg C)$. As mentioned earlier, in the following sections, we shall use the *rules* to represent absorbable axioms in a TBox to emphasize one-way reasoning.

Chapter 2

Description Logics

Up to now we have been working with informal DL, i.e., the person who uses it has to know the meaning of the symbols in it. We now re-express all of the above in more formal terms and then move to introduce formal description logic systems, i.e., the person using the system no longer has to know the meaning of the symbols in the language or the meaning behind any rule — the only thing communicated is the form of the expressions and rules, and the relations between them [13]. At last, we will introduce the automatic reasoning procedure for a formal DL based system.

2.1 The Language of Description Logics

Similarly to defining a formal language, the description logic language $\mathcal{ALCN}$ can be defined as follows:

Definition 2.1 ($\mathcal{ALCN}$ Description Logic Language)
The $\mathcal{ALCN}$ description language is based on two components: ***alphabet*** *and* ***grammar****.*

The ***alphabet*** *consists of the following sets:*

(i) A set of atomic concepts NC.

(ii) A set of atomic roles NR.

(iii) A set of punctuation symbols $\{(,)\}$

(iv) A set of connective symbols $\{\neg, \sqsubseteq, \equiv, \sqcup, \sqcap, \Rightarrow\}$

(v) A set of role restriction connective symbols $\{\forall, \exists, (\geq\ n), (\leq\ n)\}$ *where* n *is a nonnegative integer.*

The ***grammar****, which defines concept or role expressions based on NC and NR, is given by:*

(i) Each of the atomic concepts is a concept based on NC.

(ii) Each of the atomic roles is a role based on NR.

(iii) If C *and* D *are concepts based on NC, and* R *a role based on NR, then* $(\neg C)$*,* $(C \sqcap D)$*,* $(C \sqcup D)$*,* $(C \sqsubseteq D)$*,* $(C \equiv D)$*,* $(C \Rightarrow D)$*,* $(\exists R.C)$*,* $(\forall R.C)$*,* $(\geq nR.C)$*,* $(\leq nR.C)$ *are all concepts.*

(iv) Nothing else is a concept or a role based on NC and NR.

Based on the above definition, it is not difficult for us to give definitions to $\mathcal{ALCN}$ extension languages by modifying the **alphabet** or **grammar** building rules. For example, if we add a new rule such as *"let* R *and* S *be a role respectively, then* $(S \sqcap R)$*,* $(S \sqcup R)$*,* $(S \sqsubseteq R)$ *are all roles"*. Thus, we have more power to express roles. More expressive description logics can be found in [2].

Now that we have an **alphabet** and a **grammar**, we call a grammatically valid logic expression a **sentence**. A **sentence** is also called an *axiom* in description logics. A set of axioms is called a *terminology* or a TBox.

In the next section we will discuss the meaning of terminologies i.e., semantics of a TBox. After that, we will discuss automatic reasoning with formal description languages.

2.2 Semantics of Description Logics

From a semantics point of view, a concept is interpreted as a *set of individuals*, while a role is interpreted as a *set of pairs of individuals*. In order to define a formal semantics of a DL language, we define an interpretation $\mathcal{I}$ that consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of interpretation) and an interpretation function, which assign every atomic concept A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every atomic role R to a binary relation $R^I \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Thus, we can give the definition for an *interpretation* of a DL language.

Definition 2.2(Interpretation) *An* interpretation *is a pair* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$*, where* $\Delta^{\mathcal{I}}$ *is a non-empty set, called the domain of* $\mathcal{I}$*, and* $.^{\mathcal{I}}$ *is a function mapping NC to* $2^{\Delta^{\mathcal{I}}}$ *and NR to* $2^{\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}}$.

An interpretation $\mathcal{I}$ *is called* admissible *with respect to a TBox* $\mathcal{T}$ *if* $\mathcal{I}$ *satisfies every axiom in* $\mathcal{T}$*, denoted by* $\mathcal{I} \models \mathcal{T}$*. We denote by the set* $Int(\mathcal{L})$ *for all admissible interpretations with respect to* $\mathcal{T}$ *of the language* $\mathcal{L}$*. If a non-empty subset for* $\mathcal{I}$ *is able to satisfy all axioms in a TBox* $\mathcal{T}$*, then we call this set a* model *of* $\mathcal{T}$*. For two arbitrary TBox* $\mathcal{T}$ *and* $\mathcal{T}'$*, for every* $\mathcal{I} \in Int(\mathcal{L})$*, if* $\mathcal{I} \models \mathcal{T}$*, we also have* $\mathcal{I} \models \mathcal{T}'$*, we call* $\mathcal{T}$ entails $\mathcal{T}'$*, denoted as* $\mathcal{T} \models \mathcal{T}'$*. Two TBox are called* equivalent*, denoted as* $\mathcal{T} \equiv \mathcal{T}'$*, iff* $\mathcal{T} \models \mathcal{T}'$ *and* $\mathcal{T}' \models \mathcal{T}$.

From the above definitions, it is easily to draw the following conclusions:

Proposition 2.1 *Let* $\mathcal{L}$ *be a DL and* $\mathcal{T}$ *a TBox. An interpretation* $\mathcal{I} \in Int(\mathcal{L})$ *is a model of* $\mathcal{T}$ *iff, for each* $C_1 \sqsubseteq C_2 \in T$*,* $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ *holds, and for each* $C_1 \equiv C_2 \in \mathcal{T}$*,* $C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$ *holds. A concept* $C \in \mathcal{L}$ *subsumes a concept* $D \in \mathcal{L}$ *with respect to* $\mathcal{T}$ *iff, for all* $\mathcal{I} \in Int(\mathcal{L})$ *with* $\mathcal{I} \models \mathcal{T}$*,* $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ *holds.*

Proposition 2.1 is also applicable to inductive concept descriptions such as:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{I} \\
\bot^{\mathcal{I}} &= \phi \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \sqcap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b \ (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b \ (a,b) \in R^{\mathcal{I}}\}
\end{aligned}
$$

Therefore, we naturally come to the following lemma:

Lemma 2.1 *The interpretation $C^{\mathcal{I}}$ of a compound concept $C \in \mathcal{L}$ depends only on the interpretation of those atomic concepts and roles that occur syntactically in C.*

Based on the semantics of a DL, we can further give definitions to the common reasoning tasks such as *satisfiability*, *subsumption*, *equivalence*, and *disjointness*.

Definitions 2.3 (satisfiability, subsumption, equivalence and disjointness):

***Satisfiability**: A concept C is* satisfiable *with respect to $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that C^I is nonempty. Otherwise, this concept is* unsatisfiable.

***Subsumption**: A concept C is subsumed by a concept D with respect to $\mathcal{T}$ if $C^I \subseteq D^I$ for every model $\mathcal{I}$ of $\mathcal{T}$.*

***Equivalence**: Two concepts C and D are equivalent with respect to $\mathcal{T}$ if $C^I = D^I$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \equiv D$.*

***Disjointness**: Two concepts C and D are disjoint with respect to $\mathcal{T}$ if $C^I \cap D^I = \phi$ for every model $\mathcal{I}$ of $\mathcal{T}$.*

From the above definitions, the relationship between *disjointness* and *satisfiability* is obvious: if two concepts are disjoint, then the conjunction of these two concepts is unsatisfiable. Notice that, in the syntax of Description Logics, concept expressions are variable-free. In fact, a concept expression denotes the set of all individuals satisfying the properties specified in the expression. Therefore, each grammatically valid axiom in description logics can be regarded as a *first-order logic sentence*. For example,

$(C \sqcap D)$ can be regarded as a first-order logic sentence $(C(x) \wedge D(x))$, where x ranges over all individuals in the domain $\mathcal{I}$ and $C(x)$ is true to those individuals that belong to concept C. As a matter of fact, by doing this, we can describe description logics as a fragment of *first-order logic*. Concepts with role restrictions can also be expressed as the fragments of *first-order logic* as well. For example:

$$\exists R.C : \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}, R(x,y) \wedge C(y)\}$$
$$\forall R.C : \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}, R(x,y) \rightarrow C(y)\}$$

Since we have proven in *first-order logic* that the expression $C(x) \rightarrow D(x)$ is equivalent to the expression $\neg C(x) \vee D(x)$, it is obvious that the above subsumption problem can be reduced to the satisfiability problem. In fact, the following proposition indicates that subsumption, equivalence, and disjointness of concepts can all be reduced to the satisfiability problem and vice versa:

Proposition 2.2 (Reduction to Unsatisfiabililty and vice versa)

For arbitrary concepts C and D

(i) $C \sqsubseteq D \iff C \sqcap \neg D$ *is unsatisfiable;*

(ii) $C \equiv D \iff$ *both* $(C \sqcap \neg D)$ *and* $(\neg C \sqcap D)$ *are unsatisfiable;*

(iii) C *disjoint* $D \iff C \sqcap D$ *is unsatisfiable;*

The proof of the above proposition is obvious. Certainly, the above statements also hold with respect to a TBox.

2.3 Reasoning with respect to TBox and ABox

2.3.1 Reduce Concept Reasoning w.r.t. a TBox to Concept Reasoning w.r.t. an Empty TBox

From the above discussion we have seen that all concept reasoning problems can be reduced to the concept satisfiability problems. Thus, it is easier to develop reasoning procedures by only considering concept satisfiability. Also, it is conceptually easier

to do reasoning based on a concept without a TBox, i.e., a concept with an empty TBox. However, in real application, concepts usually come in the context of a TBox. Let us consider if it is possible to reduce concept reasoning w.r.t. a TBox to concept reasoning w.r.t. an empty TBox.

In general, TBox axioms of $\mathcal{ALCN}$ DL have the form:

$$C \sqsubseteq D \qquad \text{or} \qquad C \equiv D$$

To do concept reasoning with respect to a TBox, in which all axioms have the above mentioned format, generally we have two ways to reduce reasoning problems w.r.t. a TBox to reasoning problems w.r.t. an empty TBox. The first method is called *unfolding*. For example, consider the following TBox:

(i)	**Parent**	$\equiv$	**Father** $\sqcup$ **Mother**
(ii)	**Mother**	$\equiv$	**Woman** $\sqcap \exists hasChild.$**Person**
(iii)	**Woman**	$\equiv$	**Female** $\sqcap$ **Person**
(iv)	**Father**	$\equiv$	**Man** $\sqcap \exists hasChild.$**Person**
(v)	**Man**	$\equiv$	**Male** $\sqcap$ **Person**

Figure 4: A TBox example

In this example we call the kind of axiom in which there is an equality whose left-hand side is an atomic concept a *complete definition axiom*. The right-hand side is called the *definition* for the left-hand side atomic concept. The left-hand side atomic concept is called a *defined concept*. If an atomic concept only occurs on the right-hand side, we call this concept a *base concept*. Accordingly, we call the set of concepts which only occur on the right-hand *base-concept set*; and the set of concepts which occur on the left-hand side a *name-concept set*. If the left-hand side of an axiom is an atomic concept A and the relation symbol with right-hand side is not the equivalence symbol but the inclusion symbol which represents only necessary conditions, we call the concept A a *primitive concept*. Please note that some papers use the notion *primitive concept* for different meaning which is called *base concept* in this thesis. Also, let A and B be atomic concepts occurring in $\mathcal{T}$. We say that A *directly uses*

B if B occurs on the right-hand side of a definition of A. We call *uses* the transitive closure of the relation *directly uses*. Then, $\mathcal{T}$ contains a cycle and is called *cyclic* accordingly, iff there exists an atomic concept in $\mathcal{T}$ that *uses* itself. Otherwise, $\mathcal{T}$ is called acyclic.

In the following part, we may assume that our TBox of interest is acyclic. For the cyclic TBox, we have two ways to work on it:

The first solution is to convert cyclic axioms into acyclic axioms. For example, we have the following axiom:

(i) $A \sqsubseteq B \sqcap (\neg A \sqcup C)$

We can rewrite this axiom as:

(ii) $\neg A \sqcup (B \sqcap (\neg A \sqcup C))$

(ii) which can be further simplified as:

(iii) $\neg A \sqcup (B \sqcap C)$

That is:

(iv) $A \sqsubseteq (B \sqcap C)$

Thus, we have converted a cyclic axiom into an acyclic axiom without changing its original semantics. However, not all cyclic axioms can be converted into acyclic axioms by rewriting. For example,

$\mathbf{Human} \sqsubseteq \forall hasParent.\mathbf{Human}$

In this case, descriptive semantics still apply with some special considerations. Please refer to [1] for more detail. If there is no explicit specification, the conclusions drawn in this thesis are all applicable to cyclic TBoxes as well.

We also assume that if an atomic concept belongs to the name-concept set in a specific TBox, then it has and only has one definition. We denote the TBox in Figure

(i) **Mother** $\equiv$ **Female** $\sqcap$ **Person** $\sqcap$ $\exists hasChild.$**Person**
(ii) **Parent**$\equiv$(**Male**$\sqcap$**Person**$\sqcap\exists hasChild.$**Person**)$\sqcup$(**Female**$\sqcap$**Person**$\sqcap$ $\exists hasChild.$**Person**)
(iii) **Woman**$\equiv$ **Female**$\sqcap$**Person**
(iv) **Father**$\equiv$ **Male**$\sqcap$ **Person**$\sqcap$ $\exists hasChild.$**Person**
(v) **Man**$\equiv$ **Male**$\sqcap$ **Person**

Figure 5: An unfolded TBox example

4 as $\mathcal{T}$. We have the TBox shown in Figure 5 by replacing the defined concepts occurring on the right-hand side by their definitions recursively.

We call the TBox generated in this way an *unfolded* TBox. In this example, we refer to the *unfolded* TBox of $\mathcal{T}$ as $\mathcal{T}'$. We have the following proposition:

Proposition 2.3 (TBox $\mathcal{T}$ and its unfolded TBox $\mathcal{T}'$)

(i) $\mathcal{T}$ *and* $\mathcal{T}'$ *have the same name-concept set and base-concept set;*

(ii) $\mathcal{T}$ *and* $\mathcal{T}'$ *are equivalent.*

The proof of this proposition is quite simple. Suppose $A \equiv C$ and $B \equiv D$ are two definition axioms in $\mathcal{T}$ such that C uses B. Let C' be the concept obtained from C by replacing B in C with D, and let $\mathcal{T}'$ be the unfolded TBox of $\mathcal{T}$ obtained by replacing $A \equiv C$ with $A \equiv C'$. Then the base-concept set and name-concept set in $\mathcal{T}$ and $\mathcal{T}'$ are identical since no new concept being introduced on the left-hand side or right-hand side of $\mathcal{T}'$ nor is any concept missing on the left-hand side or right-hand side of $\mathcal{T}'$. In addition, these two TBoxes are equivalent since we replaced a concept by its equal concept. Then the two TBoxes have the same models.

Now let us turn our attention back to the reasoning problem w.r.t. TBoxes. Suppose we need to check the satisfiability of the concept K: (**Woman**$\sqcap\neg$ **Father**) with respect to the TBox $\mathcal{T}$ in Figure 4. Because $\mathcal{T}$ is equivalent to $\mathcal{T}'$ in Figure 5, the satisfiability checking for K w.r.t. $\mathcal{T}$ is equivalent to the satisfiability checking w.r.t. $\mathcal{T}'$. We replace all concepts in the concept of interest with its definitions in the unfolded TBox:

(i) (**Woman**$\sqcap\neg$**Father**) w.r.t. $\mathcal{T} \Longrightarrow$

(ii) **Female**$\sqcap$**Person**$\sqcap(\neg$**Male**$\sqcup\neg$**Person**$\sqcup\forall hasChild.\neg$**Person**$)$

Note that in axiom (i), we keep the "w.r.t. $\mathcal{T}$". However, in axiom (ii), we removed "w.r.t. $\mathcal{T}$" because every restriction in TBox $\mathcal{T}^{'}$ has been recursively applied. In this way, we have unfolded a TBox and reduced the concept reasoning with respect to a TBox to concept reasoning with an empty TBox.

Another way of reducing concept reasoning w.r.t. a TBox to concept reasoning with an empty TBox is called *generalization*.

Before we start to discuss *generalization*, let us first summarize TBox semantics. As we have already discussed, a TBox $\mathcal{T}$ usually consists of a set of axioms in the form of $(C \sqsubseteq D)$ or $(C \equiv D)$, where C and D are concepts. An interpretation $\mathcal{I}$ satisfies $\mathcal{T}$ if for every axioms $(C \sqsubseteq D) \in \mathcal{T}$, $(C^I \subseteq D^I)$, and for every $(C \equiv D) \in T$, $(C^I = D^I)$; $\mathcal{T}$ is *satisfiable* if there exists some non-empty interpretations that satisfies $\mathcal{T}$. In addition, by using the following equivalences, all axioms in a TBox can be reduced to axioms in the form of $\top \sqsubseteq C$ according to our previous discussion:

$$C \sqsubseteq D \Longleftrightarrow \top \sqsubseteq \neg C \sqcup D$$
$$C \equiv D \Longleftrightarrow \top \sqsubseteq (C \sqcup \neg D) \sqcap (\neg C \sqcup D)$$

Figure 6: Equivalences for generalization

Let $\mathcal{T}$ be a TBox, and a_1 and a_2 be axioms of $\mathcal{T}$. Suppose $\mathcal{M}$ is a model of $\mathcal{T}$, x is an individual and $x \in \Delta^{\mathcal{M}}$. Then x satisfies both a_1 and a_2 according to our previous discussion. Since a_1 and a_2 can be reduced to the form of $\top \sqsubseteq C_1$ and $\top \sqsubseteq C_2$, we can rewrite these two axioms into one axiom where x satisfies the axiom $\top \sqsubseteq C_1 \sqcap C_2$. Similarly, if x satisfies the axiom $\top \sqsubseteq C_1 \sqcap C_2$, we can also prove that x satisfies both $\top \sqsubseteq C_1$ and $\top \sqsubseteq C_2$. Therefore, we illustrate that an arbitrary TBox can be reduced to an axiom in the form of $\top \sqsubseteq C$, where C is called *the reduction concept* of $\mathcal{T}$. We call the procedure to achieve the *reduction concept* of a TBox as "*generalization*".

Till now we are able to reduce an arbitrary TBox $\mathcal{T}$ into $\mathcal{T}'$, where TBox $\mathcal{T}'$ has the form $\top \sqsubseteq C$. In addition, we have already shown that all concept reasoning problems can be reduced to concept satisfiability problems. To keep the generality, we need to check the satisfiability of concept E with respect to $\mathcal{T}$. We make the following proposition:

Proposition 2.4 (Generality)

C is the *reduction concept* of $\mathcal{T}$. Concept E with respect to $\mathcal{T}$ is satisfiable iff concept $E \sqcap C$ is satisfiable.

Proof

> We prove (i) first. We consider the only if direction first. Suppose concept E is satisfiable with respect to $\mathcal{T}$, a non-empty set $\mathcal{M}$ is an arbitrary model of it. If $x \in \mathcal{M}$ is an arbitrary individual of $\mathcal{M}$, then x satisfies both E and $\mathcal{T}$. Thus, x also satisfies the *reduction concept* of $\mathcal{T}$ which is C. We conclude that x satisfies both concept E and C, i.e., x satisfies $E \sqcap C$. Therefore, $\mathcal{M}$ is also a model of $E \sqcap C$.

Just as we have seen, similar to unfolding, we can also apply the *generalization* to convert a problem of concept reasoning with respect to a TBox to a problem of concept reasoning with an empty TBox.

2.3.2 Reduce Arbitrary Reasoning to ABox Reasoning

In general, a knowledge-base consists of two parts: a terminology set called TBox and an assertion set called ABox. In the previous discussions, we mainly focused on reasoning tasks with a TBox. In this section, we will discuss the reasoning tasks in more general — reasoning with respect to both TBox and ABox.

We recall that an ABox contains two kinds of assertions: concept assertion in the form of $C(a)$ and role assertion in the form of $R(a, b)$. It is obvious that an ABox has

to be consistent. For example, if an ABox contains the assertions **Male**(*Tomcat*) and **Female**(*Tomcat*), the system should be able to find that, together with the TBox $\{\top \sqsubseteq \neg\mathbf{Male} \sqcup \neg\mathbf{Female}\}$, these statements are inconsistent.

In terms of model theoretic semantics, we can easily give a formal definition of ABox consistency:

Definition 2.3 (ABox consistency) *An ABox $\mathcal{A}$ is* consistent *with respect to a TBox $\mathcal{T}$, if there exists an interpretation that is a model of both $\mathcal{A}$ and $\mathcal{T}$.*

If $\mathcal{A}$ is *consistent* with respect to an empty TBox, we simply say that $\mathcal{A}$ is *consistent*. For example, the set of assertion {**Male**(*Tomcat*), **Female**(*Tomcat*)} is consistent with respect to an empty TBox, because without any further restrictions on the interpretation of **Male** and **Female**, the two concepts can be interpreted in such a way that they have a common element. However, the assertions are not consistent with respect to the TBox $\{\top \sqsubseteq \neg\mathbf{Male} \sqcup \neg\mathbf{Female}\}$, since in every model of it, **Male** and **Female** are interpreted as disjoint concepts.

Similar to the case of concept satisfiability checking w.r.t. a TBox, which can be reduced to concept satisfiability checking w.r.t. an empty TBox, checking the consistency of an ABox w.r.t. a TBox can also be reduced to checking an expanded ABox w.r.t. an empty TBox. We denote the expansion of ABox $\mathcal{A}$ with respect to TBox $\mathcal{T}$ as $\mathcal{A}'$ that is obtained from $\mathcal{A}$ by replacing each concept assertion $C(a)$ in $\mathcal{A}$ with the assertion $C'(a)$, where C' is the expansion of C with respect to $\mathcal{T}$. As we have discussed, for every model of $\mathcal{T}$, a concept C with respect to $\mathcal{T}$ is interpreted in the same way as its expansion C'. Thus, $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff $\mathcal{A}'$ is consistent. The prototypical ABox inference task on which all other ABox inference tasks are based is instance checking, i.e., to check if an assertion is entailed by an ABox. Let $\mathcal{A}$ be an ABox, and α be an assertion. If every interpretation that satisfies $\mathcal{A}$, i.e. every model of $\mathcal{A}$, also satisfies α, we say that α is entailed by $\mathcal{A}$, denoted as $\mathcal{A} \models \alpha$.

Thus, we have the following proposition:

Proposition 2.5 (ABox and its expansion)

(i) $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff its expansion $\mathcal{A}'$ is consistent.

(ii) For a consistent ABox $\mathcal{A}$, $\mathcal{A} \models C(a)$ iff $\{\mathcal{A} \wedge \neg C(a)\}$ is inconsistent, where a is an arbitrarily chosen individual name.

Proof

We have already proved (i). Now let us prove (ii).

For the if direction to be inconsistent, let b be an arbitrary individual that satisfies $\mathcal{A}' = \{\mathcal{A} \wedge \neg C(b)\}$. $\mathcal{A}'$ is comprised of two parts: $\mathcal{A}$ and $\neg C(b)$. Since $\mathcal{A}$ is consistent, then $\neg C(b)$ must be inconsistent due to the inconsistency of $\mathcal{A}'$. That is, the assertion $C(b)$ is consistent. Since b is an arbitrary individual, we conclude that $\mathcal{A} \models C(a)$.

The only if direction is obvious.

Furthermore, we have the following lemma:

Lemma 2.2(Reduce concept reasoning to ABox reasoning)

Concept C is satisfiable iff $C(a)$ is consistent, where a is an arbitrary individual name.

From model theory, the above proposition is obvious because if C is satisfiable, then there must exist a model for it. An arbitrary individual from its model also makes $C(a)$ consistent. In addition, it is obvious that the set of individuals which satisfy $C(a)$ is a model of C.

From **Lemma 2.2**, we are able to draw the conclusion that all reasoning tasks in a DL based KB system can be reduced to ABox reasoning tasks. To generalize and simplify our discussion, in the following sections, we will mainly focus our discussion on the reasoning tasks with respect to a TBox. As we have just seen, all our discussion results and conclusions are still applicable to general cases with respect to both TBox

and ABox.

Since our discussion will focus on concept reasoning with respect to a TBox (we also use the term *reasoning with a TBox*), we will introduce a frequently used technique — *TBox rewriting* in the following sections, namely the discussion about TBox equivalence.

2.3.3 TBox Equivalence

In practice, we are often confronted with the problem to check if two TBoxes are in fact semantically equivalent. Especially, *TBox rewriting* is often used to improve reasoning performance. Therefore, the *TBox* equality checking is one of the topics of our interest.

From our previous discussion, we know that each TBox can be reduced by *generalization* to a concept. The resulting concept is called the *reduction concept of* $\mathcal{T}$. It is obvious that if two *reduction concepts* are equivalent, then the corresponding two TBoxes are also equivalent. Thus, we have the following proposition:

Proposition 2.6 *TBox* $\mathcal{T}_1$ *and TBox* $\mathcal{T}_2$ *are equivalent iff the corresponding reduction concepts* C_1 *and* C_2 *are equivalent.*

We have already stated the proof for this proposition — each TBox can be reduced to a concept without modifying its semantics (refer to Section 2.3.1). The equivalence of the two reduction concepts represents the equivalence of the two TBoxes.

However, to compare the equivalency of two *reduction concepts* is time consuming and cumbersome from practical point of view due to requirement of semantic reasoning. Thus, to test the equivalence of two TBoxes, other indirect techniques are often applied. To introduce these techniques, let us first introduce the definition of *direct consequence* which is widely used in many logic related fields.

Definition 2.4 (Direct Consequence)

If $A \Rightarrow B$, then B is called a direct consequence *of A.*

From this definition, we have:

Proposition 2.7 *For arbitrary TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, if every axiom in $\mathcal{T}_2$ is a direct consequence of $\mathcal{T}_1$, then $\mathcal{T}_1 \models \mathcal{T}_2$.*

Proof

Given an arbitrary interpretation $\mathcal{I}$ of $\mathcal{T}_1$, if $\mathcal{I} \models \mathcal{T}_1$, then $\mathcal{I}$ satisfies all axioms in $\mathcal{T}_1$. Because each axiom in $\mathcal{T}_2$ is a direct consequence of axioms in $\mathcal{T}_1$, $\mathcal{I}$ also satisfies each axiom in $\mathcal{T}_2$. Thus, $\mathcal{I} \models \mathcal{T}_2$ also holds. From **Definition 2.2**, we have that $\mathcal{T}_1 \models \mathcal{T}_2$ hold.

By **Definition 2.2**, if $\mathcal{T}_1 \models \mathcal{T}_2$ and $\mathcal{T}_2 \models \mathcal{T}_1$, then $\mathcal{T}_2 \equiv \mathcal{T}_1$. In practice, this method is more widely used to check TBox equivalence than the method to check the reduction concepts directly.

In the following section, we will introduce an algorithm known as *Semantic Tableau Algorithm* to demonstrate in detail how to reduce reasoning tasks with respect to a TBox to reasoning tasks with respect to an ABox.

2.4 Tableau Algorithm

Before describing a tableau-based satisfiability algorithm for $\mathcal{ALC}$ in more detail, we shall illustrate the underlying ideas with some examples. Suppose in the following discussion A and B are concept names, and R and S are role names.

For the first example, assume that we want to know if $(\exists R.A) \sqcap (\exists R.B)$ is subsumed by $\exists R.(A \sqcap B)$. Then, we need to check whether the following concept

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

is unsatisfiable.

First, we shall convert the expression to negation normal form, i.e., negation signs occur only in front of concept names. As a result, we get the following concept:

$$C_0 = (\exists R.A) \sqcap (\exists R.B) \sqcap \forall R.(\neg A \sqcup \neg B))$$

Second, we try to construct an interpretation $\mathcal{I}$ such that $C_0^{\mathcal{I}} \neq \phi$. It means that there must exist an individual in $\mathcal{I}$ that is an instance of $C_0^{\mathcal{I}}$.

The algorithm generates such an individual, say b, and imposes the constraint $b \in \mathcal{C}_0^{\mathcal{I}}$ on it. Since C_0 is the conjunction of concept descriptions, b needs to satisfy the following three constraints at the same time:

(1) $b \in (\exists R.A)^{\mathcal{I}}$;

(2) $b \in (\exists R.B)^{\mathcal{I}}$;

(3) $b \in (\forall R.(\neg A \sqcup \neg B))^{\mathcal{I}}$.

From (1), we can deduce that there must exist an individual c such that $(b, c) \in R^{\mathcal{I}}$ and $c \in A$. Analogously, $b \in (\exists R.B)^{\mathcal{I}}$ implies there exist an individual d such that $(b, d) \in R^{\mathcal{I}}$ and $d \in B$. Please note that b and d do not necessarily have to be identical according to the semantics of (1) and (2). Thus,

> *For an existential restriction the algorithm may introduce a new individual as role successor if there is no role successor exists, and this individual must satisfy the constraints expressed by the restriction* [1].

Since b must also satisfy the restriction $\forall R.(\neg A \sqcup \neg B)$, and c, d are introduced as *R successors* of b, we get the additional constraints: $c \in (\neg A \sqcup \neg B)$ and $d \in (\neg A \sqcup \neg B)$. Analogously, we can deduce the rules of applying disjunction and conjunction:

> *For disjunction constraints, the algorithm tries both possibilities in successive attempts. It must backtrack if it reaches an obvious contradiction. For conjunction, the algorithm tries all conjunct elements at one time.*

(i)**The $\sqcap$-rule**
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but it does not contain both $C_1(x)$ and $C_2(x)$.
Action: $\mathcal{A}_0 = \mathcal{A} \cup \{C_1(x), C_2(x)\}$

(ii)**The $\sqcup$-rule**
Condition: $\mathcal{A}$ contains $(C_1 \cup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$.
Action: $\mathcal{A}_0 = \mathcal{A} \cup \{C_1(x)\}, \mathcal{A}_1 = \mathcal{A} \cup \{C_2(x)\}$

(iii)**The $\exists$-rule**
Condition: $\mathcal{A}$ contains $(\exists R.C)(x)$ if there is no individual name z such that $C(z)$ and $R(x, z)$ are in $\mathcal{A}$
Action: $\mathcal{A}_0 = \mathcal{A} \cup \{C(z), R(x, z)\}$

(iv)**The $\forall$-rule**
Condition: $\mathcal{A}$ contains $(\forall R.C)(x)$ and $R(x, y)$, but it does not contain $C(y)$.
Action: $\mathcal{A}_0 = \mathcal{A} \cup \{C(y)\}$

Figure 7: *Tableau Algorithm* rules for $\mathcal{ALC}$

Let $\mathcal{A}$ be an ABox and $\mathcal{A}_0$, $\mathcal{A}_1$ be the tableau expansions of $\mathcal{A}$. The *tableau-based* satisfiability algorithm for $\mathcal{ALC}$ is summarized in Figure 7.

For other rules such as number restriction rules, please refer to [1].

To simplify our discussion with *Tableau-algorithms* and to understand how it is used in applications, we will introduce some definitions.

Note that a TBox can be restricted to contain only the inclusion axioms and equality axioms by using the equivalence shown in Figure 6. Therefore, without loosing generality, we can restrict our discussion to axioms in the form of $C_1 \sqsubseteq C_2$ or $C_1 \equiv C_2$, where $C_i \in \mathcal{L}$.

In practice, all DL reasoning problems are usually reduced to *ABox* reasoning problems by applying **Lemma 2.2**. Therefore, a tableau expansion is usually applied in the way with respect to a specific concept. To facilitate our following discussions, we will introduce the notion of a *witness* as an abstraction of the *Tableau Algorithms* applied to a specific concept.

Definition 2.5 (witness) *Let $\mathcal{L}$ be a DL and $C \in \mathcal{L}$ be a concept. A witness $\mathcal{W} = \{\Delta^{\mathcal{W}}, \cdot^{w}, \mathcal{L}^{w}\}$ for C consists of a non-empty set $\Delta^{\mathcal{W}}$, a function $\cdot^{w}$ that maps NR to*

$2^{\Delta^{\mathcal{W}} \times \Delta^{\mathcal{W}}}$, *and a function* $\mathcal{L}^{\mathcal{W}}$ *that maps* Δ^{W} *to* $2^{\mathcal{L}}$ *such that the following properties hold:*

($\mathcal{W}_1$*) there exists some* $x \in \Delta^{\mathcal{W}}$ *with* $C \in \mathcal{L}(x)$*;*

($\mathcal{W}_2$*) there exists an interpretation* $\mathcal{I} \in Int(\mathcal{L})$ *that stems from* $\mathcal{W}$*;*

($\mathcal{W}_3$*) for each interpretation* $\mathcal{I} \in Int(\mathcal{L})$ *that stems from* $\mathcal{W}$*, it holds that* $D \in \mathcal{L}^{\mathcal{W}}(x)$ *implies* $C \sqsubseteq D$*.*

An interpretation $\mathcal{I}$ *is said to stem from* $\mathcal{W}$ *if it satisfies:*

(i) $\Delta^{\mathcal{I}} = \Delta^{\mathcal{W}}$

(ii) $\cdot^{\mathcal{I}} |_{NR} = \cdot^{\mathcal{W}}$*, and*

(iii) for each $A \in \mathcal{NC}$*,* $(A \in \mathcal{L}^{\mathcal{W}}(x) \Longleftrightarrow x \in A^{\mathcal{I}})$ *and* $(\neg A \in \mathcal{L}^{\mathcal{W}}(x) \Longleftrightarrow x \notin A^{\mathcal{I}})$*.*

A witness $\mathcal{W}$ *is called admissible w.r.t. a TBox* $\mathcal{T}$ *if there exists an interpretation* $\mathcal{I} \in Int(\mathcal{L})$ *that stems from* $\mathcal{W}$ *with* $\mathcal{I} \models \mathcal{T}$*.*

Please note that, for any witness $\mathcal{W}$, ($\mathcal{W}2$) and (iii) of "stemming" implies that there can not exist $x \in \Delta^{\mathcal{W}}$ and $A \in NC$, such that $\{A, \neg A\} \in \mathcal{L}^{w}(x)$. In general, however, more than one interpretation may stem from a witness. Suppose there are two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ that stem from $\mathcal{L}(x)$. It might be true that there exists an element $x \in \Delta^{\mathcal{W}}$ and $\mathcal{L}^{w}(x) \wedge \{\mathcal{A}, \neg\mathcal{A}\}$ holds because $x \in A^{\mathcal{I}_2}$ and $x \in \neg A^{\mathcal{I}_1}$ hold at the same time. Thus, in order to simplify our discussion, we only focus on one special witness which stems from only one particular interpretation. Each interpretation stems only one particular witness, we call such a witness *canonical witness*.

Definition 2.6 (Canonical Witness)

Let $\mathcal{L}$ *be a DL. For any interpretation* $\mathcal{I} \in Int(\mathcal{L})$ *where* $Int(\mathcal{L})$ *is the set of all interpretation, we define the canonical witness* $\mathcal{W}_{\mathcal{I}} = \{\Delta^{\mathcal{W}_{\mathcal{I}}}, \cdot^{\mathcal{W}_{\mathcal{I}}}, \mathcal{L}^{\mathcal{W}_{\mathcal{I}}}\}$ *as follows:*

(i) $\Delta^{\mathcal{W}_{\mathcal{I}}} = \Delta^{\mathcal{I}}$

(ii) $\cdot^{\mathcal{W}_I} = \cdot^{\mathcal{I}}|_{NR}$*, and*

(iii) $\mathcal{L}^{\mathcal{W}_{\mathcal{I}}} = \{\forall x \in \Delta^{\mathcal{W}_{\mathcal{I}}} \mid \{D \in \mathcal{L} \Rightarrow x \in D\}\}$

The following elementary properties of a canonical witness will be useful in our following discussions.

Proposition 2.8(properties of a canonical witness)
Let $\mathcal{L}$ be a DL, $C \in \mathcal{L}$, and $\mathcal{T}$ a TBox. For each $\mathcal{I} \in Int(\mathcal{L})$ with $C^{\mathcal{I}} \neq \phi$,

(i) each interpretation $\mathcal{I}'$ stemming from $\mathcal{W}_{\mathcal{I}}$ is isomorphic to $\mathcal{I}$.

(ii) $\mathcal{W}_{\mathcal{I}}$ is a witness for C,

(iii) $\mathcal{W}_{\mathcal{I}}$ is an admissible witness w.r.t. $\mathcal{T}$ iff $\mathcal{I} \models \mathcal{T}$.

Proof

(i) Let $\mathcal{I}_1$ stem from $\mathcal{W}_{\mathcal{I}}$. This implies $\Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{I}_1} = \cdot^{\mathcal{I}}|_{NR}$. For each $x \in \Delta^{\mathcal{I}_1}$, and any concept D, we also have $x \in \Delta^{\mathcal{I}}$ because $\Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}}$. Thus, if $D \in \mathcal{L}^{\mathcal{W}_{\mathcal{I}_1}}$, we have $x \in D$. This implies $D \in \mathcal{L}^{\mathcal{W}_{\mathcal{I}}}$. This proves that $\cdot^{\mathcal{I}_1}|_{NC} = \cdot^{\mathcal{I}}|_{NC}$ and hence $\mathcal{I}_1$ and $\mathcal{I}$ are isomorphic.

(ii) Properties ($\mathcal{W}1$) and ($\mathcal{W}2$) hold according to the definition of $\mathcal{W}_{\mathcal{I}}$. Obviously, $\mathcal{I}$ stems from $\mathcal{W}_{\mathcal{I}}$ and from (i) it follows that each interpretation $\mathcal{I}'$ stemming from $\mathcal{W}_{\mathcal{I}}$ is isomorphic to $\mathcal{I}$, hence ($\mathcal{W}3$) holds.

(iii) Since $\mathcal{I}$ stems from $\mathcal{W}_{\mathcal{I}}$, $\mathcal{I} \models \mathcal{T}$ implies that $\mathcal{W}_{\mathcal{I}}$ is admissible according to the definition of an admissible witness. For the only if direction: if $\mathcal{W}_{\mathcal{I}}$ is admissible w.r.t. $\mathcal{T}$, then there is an interpretation $\mathcal{I}^*$ stemming from $\mathcal{W}_{\mathcal{I}}$ with $\mathcal{I}^* \models \mathcal{T}$. Since $\mathcal{I}$ is isomorphic to $\mathcal{I}^*$, hence $\mathcal{I} \models \mathcal{T}$.

As a consequence, we are able to conclude the existence of an admissible witness is closely related to the satisfiability of a concept w.r.t. a TBox. The relationship is described by the following proposition:

Proposition 2.9 *Let $\mathcal{L}$ be a DL. A concept $C \in \mathcal{L}$ is satisfiable w.r.t. a TBox $\mathcal{T}$ iff it has a witness that is admissible w.r.t. $\mathcal{T}$.*

Proof

Let us consider the "⇒" direction first. Let $\mathcal{W}$ be the admissible witness w.r.t. $\mathcal{T}$, $\mathcal{I}$ is the interpretation with $\mathcal{I} \models \mathcal{T}$. We have $C^{\mathcal{I}} \neq \phi$. Thus, C is satisfiable.

Then, let us consider the "⇐" direction. Since C is satisfiable, there exists an interpretation $\mathcal{I} \in Int(\mathcal{L})$ which makes $C^I \neq \phi$ w.r.t. $\mathcal{T}$. That is $\mathcal{I} \models T$. Let the corresponding canonical witness of C be $\mathcal{W}_{\mathcal{I}}$. Then, $\mathcal{W}_{\mathcal{I}}$ is admissible.

2.5 Complexity of Reasoning

It is much appealing to apply tableau-based algorithms for solving reasoning problems in a DL knowledge-based system due to its directness. However, when applying the expansion formula described by Figure 6, disjunctions are added to the label of each node of the tableau for each general axiom (one disjunction is added for axioms in the form of $C_1 \sqsubseteq C_2$; two disjunctions are added for axioms in the form of $C_1 \equiv C_2$). This leads to an exponential increase in the search space as the number of nodes and axioms increases. This kind of increase will directly cause non-determination for reasoning tasks. For example, consider a KB with 100 inclusive axioms in the form of $C_1 \sqsubseteq C_2$. Such a KB is really small compared to knowledge-based systems in real application. However, in the worst-case, by directly applying tableau-expansion, there could be 100 disjunctions to be added to the tableau, and those disjunctions can be non-deterministically decomposed in 2^{100} different ways[1]. We have to find a more efficient way to do reasoning. In other words, the reasoning algorithm based on Semantic Tableau has to be optimized to meet the requirement of real applications. In the following sections, we will mainly focus on a technique called "TBox rewriting" to fulfill the optimization. The goal is to rewrite a TBox to a format suited for faster reasoning.

[1] To apply Tableau rules, we have to convert the axiom in the form of $C \sqsubseteq D$ into $\neg C \sqcup D$ to apply the disjunction rule.

Chapter 3

Normalization and Simplification

It is obvious that getting rid of redundant knowledge from a KB can improve the system's reasoning performance. We call the procedure of removing redundant knowledge *simplification* in this thesis. In addition, when we talk about reasoning in knowledge-based system, we are in fact talking about automatic reasoning using a computer. That is, all forms and expressions that we are concerned about must be suitable for automatic processing with computers. For example, the logic expressions $(A \sqcap (B \sqcap (C \sqcup D))$, $(A \sqcap B \sqcap (C \sqcup D))$ and $((A \sqcap B \sqcap C) \sqcup (A \sqcap B \sqcap D))$ are hard to be recognized as identical by a syntactical comparison. To solve such kind of problem, one solution is to convert all logic expressions into the same format before reasoning. In other words, those expressions need to be normalized before reasoning. The procedure of computing normalized expression is called *normalization*. Based on the above discussion, it is not hard to find that simplification and normalization are in fact two techniques tightly related — it is obvious that a normalized expression facilitates to find syntactical equivalence, contradictions, tautologies, and can be simplified easier. In this section, we will briefly introduce the normalization and simplification techniques used in our system. The result and conclusions of this chapter will be directly used in the following sections.

3.1 Normalization

3.1.1 Enhanced Negation Normal Form

In real KB applications, especially those KBs manually constructed, axioms may not be in a same format. Those axioms may be more readable for humans, but it might be hard to automatically find obvious tautologies and contradictions. To solve this problem, usually we employ the NNF (negation normal form) to normalize the logic expressions to facilitate detection of conflicts. However, in many cases using NNF is still not enough to find direct equalities or contradictions. In the above mentioned example, even though all expressions are in NNF, they still can not be directly recognized syntactically to be equivalent.

Some Reasoners, such as RACER, use an enhanced NNF to improve the search performance [6]. The steps of normalizing an arbitrary logic expression into enhanced NNF can be explained as follows.

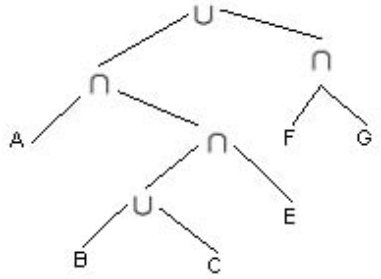

Figure 8: An example of an NNF syntax tree

The first step is to draw the logic hierarchy for an NNF expression. Since the steps of converting an arbitrary logic expression into NNF is trivial, we shall not discuss it here. To draw the logic hierarchy, we put operands under operators and expand the operands inductively. For example, the syntax tree of the logic expression $(A \sqcap ((B \sqcup C) \sqcap E)) \sqcup (F \sqcap G)$ in NNF is shown in Figure 8.

The next step is to convert NNF into enhanced NNF. The primary idea is to absorb the lower level operator if the lower level operator is the same as its upper

level operator since keeping the same logic operator in two adjacent level is in fact redundant. Thus, the enhanced NNF format of the above mentioned logic expression is shown in Figure 9.

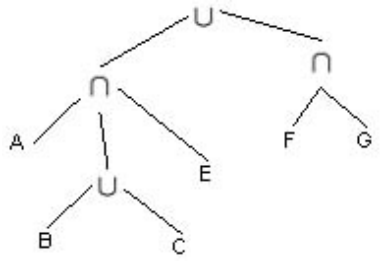

Figure 9: Conversion of NNF into enhanced NNF

In many cases, the enhanced NNF can help to reduce the search space. However, in some cases, the enhanced NNF format is still not enough to help dectect obvious contradictions or equalities directly. For example, the two logic expressions $(A \sqcap (B \sqcup C) \sqcap E) \sqcup (F \sqcap G)$ and $(F \sqcup (A \sqcap E \sqcap (B \sqcup C))) \sqcap (G \sqcup (A \sqcap E \sqcap (B \sqcup C))))$ are syntactically equivalent by applying De Morgen's law. Unfortunately, even though both of these expressions are already in enhanced NNF format, they can not directly be found as syntactically equivalent.

In addition, by using the enhanced NNF format to deal with our logic storage model, we have to maintain a tree-like data structure as an internal data model. For such kind of data model some operations such as searching are complicated.

3.1.2 Normalize TBox into Set and Logic operations into Set Operations

To find a suitable model for logic expressions, a solution is to implement logic operations as set operations and convert tree-like data structures into flat sets. Notice that the relationship between axioms in a TBox is in fact a conjunction. For example, let α_1 and α_2 be two axioms in $\mathcal{T}$. Then, $\alpha_1 \sqcap \alpha_2 \in \mathcal{T}$. Moreover, from the above discussion we have seen that each axiom in a TBox can be converted into the format $\top \sqsubseteq C$, where C is a concept. Since "$\top \sqsubseteq$" exists in each axiom, there is neither the

need to store it into our physical data structure nor for the conjunction relationship. Therefore, we conclude that *a TBox can be normalized into a set of concepts.*

Now let us consider the attributes of an axiom in a TBox. Suppose we have an axiom $\top \sqsubseteq C \sqcap D$ in a TBox. As we know, this axiom can be split into two axioms $\top \sqsubseteq C$ and $\top \sqsubseteq D$ without changing the semantics of the original TBox. In fact, we can convert an arbitrary concept into CNF (conjunction normal form), i.e., in the resulting concept, there are only two levels: the first level only contains conjunctions and the second level only contains disjunctions. In this way, we can easily remove conjunctions from TBox axioms. Therefore, *each axiom in a TBox can be normalized to a kind of axiom not containing conjunctions.*

Based on this idea, in the above example both $(A \sqcap (B \sqcup C) \sqcap E) \sqcup (F \sqcap G)$ and $(F \sqcup (A \sqcap E \sqcap (B \sqcup C))) \sqcap (G \sqcup (A \sqcap E \sqcap (B \sqcup C))))$ can be easily normalized to the set $(F, A), (F, E), (F, B, C), (G, A), (G, E), (G, B, C)$. Thus, they are obviously equivalent, shown simply by a syntactic checking.

3.2 Simplification

In addition to normalization, we can also include a series of simplifications during TBox rewriting so that syntactically obvious contradictions and tautologies can be detected. For example, $(A \sqcup \neg A)$ could be simplified to $\top$. The following list enumerates the most frequently used simplification formulas:

(1)$(A \sqcap \neg A) \equiv \bot$

(2)$(A \sqcup \neg A) \equiv \top$

(3)$A \sqcap (A \sqcup B) \equiv A$

(4) $A \sqcup (A \sqcap B) \equiv A$

(5)$A \sqcap (\neg A \sqcup B) \equiv A \sqcap B$

(6)$A \sqcup (\neg A \sqcap B) \equiv A \sqcup B$

(7) $(\forall R.A \sqcap \forall R.B) \equiv \forall R.(A \sqcap B)$

(8) $(\exists R.A \sqcup \exists R.B) \equiv \exists R.(A \sqcup B)$

(9) $\exists R.\ \bot \equiv \bot$

(10) $\forall R.\top \equiv \top$

(11) $(A \Rightarrow C \text{ and } B \Rightarrow C) \Longleftrightarrow ((A \sqcup B) \Rightarrow C)$

The proof for most of the above formulas is trivial. For example, formula (3) and (4) are obvious by enumerating truth table of the both sides. The others are also obvious by applying Tableau expansions or De Morgan's law.

The above simplification rules are applicable to all concepts or axioms in a TBox. For example, if a TBox contains both the axioms ($A \sqcup B$) and ($A \sqcup B \sqcup E \sqcup \exists R.M$), then the latter axiom can be directly discarded according to formula (3). Moreover, the above mentioned simplifications can be more easily detected in a normalized TBox.

Chapter 4

Absorption

4.1 Lazy Unfolding

In previous sections, we have discussed TBox reasoning and fundamental rewriting techniques such as *normalization* and *simplification*. Starting from this chapter, we will focus on some techniques that can deal more efficiently with TBox reasoning.

As we have discussed previously, the complexity of a DL based KB system reasoning is usually due to the reasoning with respect to a TBox. We have also discussed that reasoning non-determinism is often caused by the axiom unfolding during tableau expansion. One of the most intuitive optimization techniques is called *lazy unfolding* — only unfolds concepts when required during the progress of subsumption or satisfiability testing [4]. In other words, an atomic concept will only be unfolded when it occurs in a tableau expansion node. For example, if a TBox $\mathcal{T}$ contains the axiom $A \Rightarrow C$, and $(A \sqcap B) \in \mathcal{L}(x)$, while applying the $\sqcap - rule$, A and B are added to $\mathcal{L}(x)$. At this point, by applying lazy unfolding, we add C to $\mathcal{L}(x)$ as well. However, if A does not occur in $\mathcal{L}(x)$, we do nothing.

More generally, *lazy unfolding* can then be described by the following additional tableau algorithm rules:

rule 1: If $A \in \mathcal{L}(x)$ and $(A \Rightarrow C) \in \mathcal{T}$
then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \sqcup \{C\}$

rule 2: If $\neg A \in \mathcal{L}(x)$ and $(\neg A \Rightarrow C) \in \mathcal{T}$
then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \sqcup \{C\}$

where $A \in NC$[1].

Compared to normal tableau rules in which many disjunctions are introduced — the main cause for a high inefficiency— the above rules are obviously more efficient since no new disjunctions will be introduced during the expansion. In addition, *lazy unfolding* only unfolds a concept when it occurs in the expansion tree which avoids to expand the entire TBox as we do during *generalization*. As a matter of fact, *lazy unfolding* has proven to be very efficient comparing to normal tableau rules [7].

As we have just noticed, rule 1 and rule 2 can only be applied to a *rule axiom* in $\mathcal{T}$ and the left-hand side of the *rule axiom* has to be an atomic concept. In real applications, it is not always the case that all axioms in an arbitrary TBox happen to satisfy these restrictions. Therefore, an intuitive optimization technique was proposed: dividing an arbitrary TBox $\mathcal{T}$ into two parts, an unfoldable part $\mathcal{T}_u$ and a general part $\mathcal{T}_g$ such that $\mathcal{T}_g = \mathcal{T} \setminus \mathcal{T}_u$. $\mathcal{T}_u$ contains only rule axioms with concept names on the left-hand side, while $\mathcal{T}_g$ contains the rest of $\mathcal{T}$. In this way, reasoning tasks with respect to a TBox can be considered as reasoning tasks with respect to $\mathcal{T}_u$ and $\mathcal{T}_g$: apply "lazy unfolding" to $\mathcal{T}_u$ and regular tableau expansion to $\mathcal{T}_g$. As we already mentioned that the reasoning performance for $\mathcal{T}_u$ can be very high, while the reasoning performance to $\mathcal{T}_g$ might be low when applying regular tableau expansion due to the introduction of many disjunctions. By considering $\mathcal{T}_u$ and $\mathcal{T}_g$, we apply the following optimization technique: if one can move axioms from $\mathcal{T}_g$ to $\mathcal{T}_u$ while keeping the semantics of $\mathcal{T}$ unchanged, one should be able to improve the reasoning performance. This technique is called *"absorption"*.

[1]We do not list the "$\equiv$" rule in this list since the axiom $A \equiv C$ can be easily rewritten as $A \Rightarrow C$ and $\neg A \Rightarrow \neg C$, and the above mentioned rules can be easily applied as well.

4.2 Absorption

As we have seen in the previous section, the primary idea of absorption is to move axioms from $\mathcal{T}_g$ to $\mathcal{T}_u$ while keeping the semantics of the new TBox identical to the original TBox. The question is how can we achieve this goal without affecting the semantics of the original TBox. In other words, we have to define criteria for a *correct absorption* and make sure that a pre-absorption and post-absorption TBox are semantically equivalent. A discussion about *correct absorption* has been made in [8] which can be summarized as follows.

Definition 4.1 (unfolded witness, correct absorption)
Let L be a DL and $\mathcal{T}$ a TBox. An absorption of $\mathcal{T}$ is a pair of TBoxes $(\mathcal{T}_u, \mathcal{T}_g)$ such that $\mathcal{T} \equiv \mathcal{T}_u \cup \mathcal{T}_g$ and $\mathcal{T}_u$ contains only rule axioms in the form of $A \Rightarrow D$ or $\neg A \Rightarrow D$, where A is an atomic concept, i.e., $A \in NC$.
For a witness $\mathcal{W}$, if for each $x \in \Delta^{\mathcal{W}}$,

$$(A \Rightarrow D \in \mathcal{T}_u \wedge A \in \mathcal{L}^W(x)) \Longrightarrow D \in \mathcal{L}^W(x)$$

$$(\neg A \Rightarrow D \in \mathcal{T}_u \wedge \neg A \in \mathcal{L}^W(x)) \Longrightarrow D \in \mathcal{L}^W(x)$$

$$C_1 \sqsubseteq C_2 \in \mathcal{T}_g \Longrightarrow (\neg C_1 \sqcup C_2) \in \mathcal{L}^W(x)$$

$$C_1 \equiv C_2 \in \mathcal{T}_g \Longrightarrow ((\neg C_1 \sqcup C_2) \sqcap (C_1 \sqcup \neg C_2)) \in \mathcal{L}^W(x)$$

then $\mathcal{W}$ is admissible w.r.t. $\mathcal{T}$. If an absorption $(\mathcal{T}_u, \mathcal{T}_g)$ satisfies the above conditions, it is called correct. *If a witness satisfies the above condition, it is called* unfolded *w.r.t. $\mathcal{T}$.*

Please note that the symbol $\Rightarrow$ is a rule expression which represents only one-way reasoning[2]. In addition, if the "with respect to" a specific TBox is clear from the context, we may omit the w.r.t. $\mathcal{T}$ in the following sections.

Unfortunately, the definition of *correct absorption* as specified above can not be applied directly to evaluate the correctness of an absorption. For example, suppose

[2]In some papers, "$\sqsubseteq$" is also employed to represent one-way reasoning.

$\mathcal{T}'$ is an absorption of *TBoxes* $\mathcal{T}$:

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$ and $\mathcal{T}_u = \emptyset$; $\mathcal{T}_g = \{A \sqsubseteq C; \neg A \sqsubseteq D\}$;
$\mathcal{T}^{'} = \mathcal{T}_u' \cup \mathcal{T}_g'$ and $\mathcal{T}_u' = \{A \Rightarrow C; \neg A \Rightarrow D\}$; $\mathcal{T}_g^{'} = \emptyset$.

Even though it is not hard to prove that $\mathcal{T} \neq \mathcal{T}^{'}$ from a semantics point of view, we are unable to conclude that the absorption $\mathcal{T}^{'}$ is not a *correct absorption* of $\mathcal{T}$ from the definition itself. Therefore, we propose the term *valid absorption* to be the criterion to evaluate the correctness of an absorption in the following sections.

Definition 4.2 (valid absorption)
Let $\mathcal{T}$ be an arbitrary TBox, and $\mathcal{T}^{'}$ be a correct absorption of $\mathcal{T}$. If $\mathcal{T}^{'} \models \mathcal{T}$ and $\mathcal{T} \models \mathcal{T}^{'}$, then $\mathcal{T}^{'}$ is called a valid absorption *of $\mathcal{T}$.*

Based on the above definition, the following absorptions are all *valid absorptions*, provided that A is an atomic concept and C, D, E are arbitrary concepts.

Proposition 4.1
Let $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$, $\mathcal{T}_u = \phi$ and $\mathcal{T}_g = \{A \sqsubseteq D\}$; $\mathcal{T}^{'} = \mathcal{T}_u^{'} \cup \mathcal{T}_g^{'}$, $\mathcal{T}_u^{'} = \{A \Rightarrow D\}$ and $\mathcal{T}_g^{'} = \phi$. Then $\mathcal{T}^{'}$ is a valid absorption of $\mathcal{T}$.

Proof

First, it is obvious that $\mathcal{T} \models \mathcal{T}^{'}$ because $\{A \sqsubseteq D\} \Longrightarrow \{A \Rightarrow D\}$. We only need to prove that $\mathcal{T}^{'} \models \mathcal{T}$. Suppose that $\mathcal{T}^{'} \not\models \mathcal{T}$ which means the expression $(A \Rightarrow D) \sqcap \neg(A \sqsubseteq D)$ is satisfiable. This expression can be rewritten to $(A \Rightarrow D) \sqcap A \sqcap \neg D$ by rewriting $(A \sqsubseteq D)$ to $(\neg A \sqcup D)$ and convert it to NNF. To make this expression satisfiable, A and $\neg D$ must be both satisfiable. However, if A is satisfiable, then D is also satisfiable according to $(A \Rightarrow D)$. This conflicts with the fact $\neg D$ is satisfiable. Thus we infer $\mathcal{T}^{'} \models \mathcal{T}$ holds.

Proposition 4.2 *Let $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$ be a TBox, $\mathcal{T}_u = \phi$ and $\mathcal{T}_g = \{A \equiv D\}$, $A \in NC$;*

$\mathcal{T}'$ is an absorption of $\mathcal{T}$, $\mathcal{T}' = \mathcal{T}'_u \cup \mathcal{T}'_g$, $\mathcal{T}'_u = \{A \Rightarrow D; \neg A \Rightarrow \neg D\}$ and $\mathcal{T}'_g = \phi$. Then $\mathcal{T}'$ is a valid absorption of $\mathcal{T}$.

Proof

The proof is similar to the proof of **Proposition 4.1**. It is obvious that $\mathcal{T} \models \mathcal{T}'$ because $\{A \equiv D\} \Longrightarrow \{A \Rightarrow D; \neg A \Rightarrow \neg D\}$. We only need to prove that $\mathcal{T}' \models \mathcal{T}$. Suppose that $\mathcal{T}' \not\models \mathcal{T}$. That is, the expression $(A \Rightarrow D) \sqcap (\neg A \Rightarrow \neg D) \sqcap (A \sqcup D) \sqcap (\neg A \sqcup \neg D)$ is satisfiable. Suppose there exists such an individual x which satisfies this expression. First, we assume x is an instance of A, i.e., $x^I \in A^I \wedge x^I \notin (\neg A)^I$. It is obvious that such an x does not exist since we can easily infer that if $x \in A$, then both $x \in D$ and $x \in \neg D$ hold from the above expression. Similarly, if we assume that $x^I \in (\neg A)^I \wedge x^I \notin A^I$. Again, we can easily to conclude that such an x does not exist either by using the same analysis as above. Thus, $\mathcal{T}' \models \mathcal{T}$ holds.

In fact, **Proposition 4.2** can be extended to apply to a more general TBox $\mathcal{T}$ where $\mathcal{T}$ contains more than one axiom in the form of $A \equiv D$.

Lemma 4.1 *Let $\mathcal{T}$ be a TBox containing axioms entirely in the form of $A_i \equiv D_i$ where $A_i \in NC$, if $i \neq j$ $A_i \neq A_j$; $\mathcal{T}' = \mathcal{T}'_u \sqcup \mathcal{T}'_g$, $\mathcal{T}'_g = \phi$, and $\mathcal{T}'_u$ consists of the axioms in the form of $\{A_i \Rightarrow D_i; \neg A_i \Rightarrow \neg D_i\}$, where $A_i \in NC$. Then $\mathcal{T}'$ is a valid absorption of $\mathcal{T}$.*

Proof

We can prove this lemma inductively. We assume an arbitrary linearization $A_1, A_2, ..., A_k$ of the "uses" partial order on the atomic concept names appearing on the left-hand side of axioms in $\mathcal{T}$ such that, if A_i uses A_j, then $i > j$ and the defining concept for A_i is D_i.

For $i = 1$, **Lemma 4.1** holds according to **Proposition 4.2**. Suppose

for $i = k - 1$ **Lemma 4.1** also holds. We need to prove that $i = k$ also holds.

Clearly $\mathcal{T}^{'} = \mathcal{T}_u^{'} \cup \mathcal{T}_g^{'}$ holds. If $i = k - 1$, then, we have $\mathcal{T}_u = \{A_1 \Rightarrow D_1, \neg A_1 \Rightarrow \neg D_1, ..., A_{k-1} \Rightarrow D_{k-1}, \neg A_{k-1} \Rightarrow \neg D_{k-1}\}$, $\mathcal{T}_g^{'} = \{A_k \equiv D_k\}$. That is, if $i = k$, $\mathcal{T} \equiv \mathcal{T}^{'} \equiv \mathcal{T}_u^{'} \cup \mathcal{T}_g^{'}$ holds. Since the A_k does not ever appear in the left-side of $\mathcal{T}_u^{'}$ and $\mathcal{T}_u^{'}$ is acyclic, adding two axioms $\{A_k \Rightarrow D_k, \neg A_k \Rightarrow \neg D_k\}$ will not affect the reasoning behavior and reasoning result of other axioms in $\mathcal{T}_u^{'}$. Moreover, from **Proposition 4.2**, converting an equivalent axiom from $\mathcal{T}_g$ to two rule axioms and move them from $\mathcal{T}_g$ to $\mathcal{T}_u$ will not change the semantics of a TBox. Thus, when $i = k$, **Lemma 4.1** also holds.

Now that we have proven that absorption can be applied to a more general TBox containing more than one axioms in the form of $A \equiv D$, and these axioms can be completely eliminated from $\mathcal{T}_g$, a question of whether absorption can be applied to an arbitrary TBox rises. To answer this question, let us consider some special cases first.

Proposition 4.3 *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a valid absorption of a TBox $\mathcal{T}$.*

1. if $\mathcal{T}^{'}$ is an arbitrary TBox, then $(\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}^{'})$ is a valid absorption of $\mathcal{T} \cup \mathcal{T}^{'}$.

2. if $\mathcal{T}^{'}$ is a TBox that consists entirely of axioms in the form of $A \sqsubseteq D$, where $A \in NC$ and neither A nor $\neg A$ occurs on the left-hand side in $\mathcal{T}_u$, then $(\mathcal{T}_u \cup \{A \Rightarrow D\}, \mathcal{T}_g)$ is a valid absorption of $\mathcal{T} \cup \mathcal{T}^{'}$.

Proof

1. Let $C \in L$ be a concept and $\mathcal{W}$ be an unfolded witness for C w.r.t. the absorption $(\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}^{'})$, this implies that $\mathcal{W}$ is unfolded w.r.t. the absorption $(\mathcal{T}_u, \mathcal{T}_g)$ because $(\mathcal{T}_u, \mathcal{T}_g)$ is smaller. Since $(\mathcal{T}_u, \mathcal{T}_g)$ is a valid absorption, there is an interpretation $\mathcal{I}$ stemming from $\mathcal{W}$ with $\mathcal{I} \models \mathcal{T}$.

Suppose $\mathcal{I} \not\models \mathcal{T}^{'}$. Then there must exist an element $x \in \mathcal{I}$ and an axiom

$(D \sqsubseteq E) \in \mathcal{T}^{'}$ such that $x \notin (\neg D \sqcup E)^{\mathcal{I}}$.

Since $\mathcal{W}$ is unfolded with respect to the absorption $(\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}^{'})$ and $(D \sqsubseteq E) \in \mathcal{T}_g \cup \mathcal{T}^{'}$, we have that $x \in (\neg D \sqcup E)^{\mathcal{I}}$. This is a contradiction to the above conclusion $x \notin (\neg D \sqcup E)^{\mathcal{I}}$. As a result, it also holds that $\mathcal{I} \models \mathcal{T}^{'}$. Therefore, $(\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}^{'}) \models (\mathcal{T} \cup \mathcal{T}^{'})$ and $(\mathcal{T} \cup \mathcal{T}^{'}) \models (\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}^{'})$, i.e., $(\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}^{'})$ is a valid absorption of $(\mathcal{T} \cup \mathcal{T}^{'})$.

2. This is obvious from lemma 4.1 since neither A nor $\neg A$ has a rule axiom in $\mathcal{T}$.

In **Proposition 4.3**, the axiom $(A \sqsubseteq D)$ in $\mathcal{T}_g$ can be absorbed to $\mathcal{T}_u$ under the condition that neither A nor $\neg A$ has a rule definition in $\mathcal{T}_u$. Is it still possible to absorb this axiom into $\mathcal{T}_u$ with the presence of either A or $\neg A$ on the left-hand side of $\mathcal{T}_u$? To answer this question, we have the following lemmas from **Proposition 4.3**:

Lemma 4.2 *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a valid absorption of a TBox $\mathcal{T}$. If $\mathcal{T}^{'}$ is a TBox that consists entirely of axioms in the form $A \sqsubseteq D$, where $A \in NC$. If A already has a rule definition in $\mathcal{T}_u$, say $A \Rightarrow C$, and $\neg A$ has no rule definition in $\mathcal{T}_u$, then $\{(\mathcal{T}_u \setminus A \Rightarrow C) \cup (A \Rightarrow (D \sqcap C)), \mathcal{T}_g\}$ is a valid absorption of $\mathcal{T} \cup \mathcal{T}^{'}$.*

The proof for this lemma is obvious. If a TBox contains two axioms $\{A \sqsubseteq C, A \sqsubseteq D\}$, they can be reduced to one axiom $A \sqsubseteq (C \sqcap D)$ that can be absorbed into the axiom $A \Rightarrow (C \sqcap D)$ according to **Proposition 4.1**.

Similarly, we can further discuss how to absorb the axiom $A \sqsubseteq D$ in $\mathcal{T}_g$ when $\neg A$ has a rule definition in $\mathcal{T}_u$.

Lemma 4.3 *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a valid absorption of a TBox $\mathcal{T}$. If $\mathcal{T}^{'}$ is a TBox that consists entirely of axioms of the form $A \sqsubseteq D$, where $A \in NC$. If $\neg A$ already has a rule definition in $\mathcal{T}_u$, say $\neg A \Rightarrow C$, and A has no rule definition in $\mathcal{T}_u$, then $\{\mathcal{T}_u \cup \{A \Rightarrow D\}, \mathcal{T}_g \cup \{C \sqcup D\}\}$ is a valid absorption of $\mathcal{T} \cup \mathcal{T}^{'}$.*

Proof

This is equivalent to prove that the TBox $\mathcal{T}$ $\{\neg A \Rightarrow C, A \Rightarrow D, \top \sqsubseteq (C \sqcup D)\}$ is equivalent to the TBox $\mathcal{T}'$: $\{A \sqsubseteq D, \neg A \sqsubseteq C\}$.

It is obvious that $\mathcal{T}' \models \mathcal{T}$ because each axiom in $\mathcal{T}$ is a direct consequence of $\mathcal{T}'$. To prove that $\mathcal{T} \equiv \mathcal{T}'$, we only need to prove $\mathcal{T} \models \mathcal{T}'$.

Suppose $\mathcal{T} \not\models \mathcal{T}'$. First we reduce the TBox $\mathcal{T}$ and $\mathcal{T}'$ into two concepts C and C' by *generalization* (refer to Section 2.3.1). Suppose $\mathcal{I}$ is an arbitrary model of $\mathcal{T}$, then there exists an $x \in \Delta^{\mathcal{I}}$ and $x \in (C \sqcap \neg C')^I$ since $\mathcal{T} \not\models \mathcal{T}'$. That is:

$x \in ((\neg A \Rightarrow C) \sqcap (A \Rightarrow D) \sqcap (C \sqcup D) \sqcap (A \sqcup \neg C) \sqcap (\neg A \sqcup \neg D) \sqcap (\neg D \sqcup \neg C))^{\mathcal{I}}$

Let us do a case analysis about x. Suppose $x \in A^{\mathcal{I}}$. Then $x \in \neg D^{\mathcal{I}}$ upon $(\neg A \sqcup \neg D)$, and $x \in D^{\mathcal{I}}$ upon $(A \Rightarrow D)$. This is a contradiction. Analogically, we can easily enumerate all possibilities of x and conclude that such an x does not exist. That is $C \sqcap \neg C'$ is unsatisfiable. therefore, $\mathcal{T} \models \mathcal{T}'$ also holds, and the above absorption is a valid absorption.

Now we will extend the discussion to the most general case where both A and $\neg A$ have rule definitions in $\mathcal{T}_u$.

Lemma 4.4 *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a valid absorption of a TBox $\mathcal{T}$, if $\mathcal{T}'$ is a TBox that consists entirely of axioms in the form of $A \sqsubseteq E$, where $A \in NC$. If A already has a rule definition in $\mathcal{T}_u$, say $A \Rightarrow C$, and $\neg A$ also has a rule definition in $\mathcal{T}_u$, say $\neg A \Rightarrow D$, then $\{(\mathcal{T}_u \setminus \{A \Rightarrow C\}) \cup \{A \Rightarrow (C \sqcap E)\}, (\mathcal{T}_g \setminus \{A \sqsubseteq E\}) \cup \{\top \sqsubseteq D \sqcup E\}\}$ to $\mathcal{T}_g$ is a valid absorption of $\mathcal{T} \cup \mathcal{T}'$.*

The proof for this lemma is similar to the proof of **Lemma 4.3**. We will not repeat it here.

From **Lemma 4.3** and **Lemma 4.4**, it is obvious to have the following lemma:

Lemma 4.5*Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a valid absorption of a TBox $\mathcal{T}$. Suppose $\{A \equiv K_1; A \equiv$*

$K_2\} \in \mathcal{T}_g$ *and there is no rule definition in* $\mathcal{T}_u$ *for either* A *or* $\neg A$*, then the absorption* $\{A \Rightarrow K_1; \neg A \Rightarrow \neg K_1\} \in \mathcal{T}_u$*;* $\{K_1 \equiv K2\} \in \mathcal{T}_g$ *is a valid absorption of* $\mathcal{T}$*.*

We also omit the proof for this lemma here because the conclusion is obvious.

4.3 Role Absorption

So far we have discussed the *Tableau Algorithm* and its extension to *lazy unfolding*. Based on *model theory*, we also discussed concept absorption. In this section, we will turn our attention to the other aspect — *roles*. Upon *model theory*, an *interpretation* is a pair $\mathcal{I}$= $\{\Delta^{\mathcal{I}}, .^{\mathcal{I}}\}$, where $\Delta^{\mathcal{I}}$ is a non-empty set, and $.^{\mathcal{I}}$ is a function mapping NC to $2^{\Delta^{\mathcal{I}}}$ and NR to $2^{\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}}$. That is, if an individual x is related to an individual y by a role R, R is a mapping from the set of individuals x to the set of individuals y. As we mentioned before, the set of individual x can be seen as a concept, say C. To denote this, we therefore use an axiom such as *Domain*(R, C). Accordingly, the set of individual y can also be seen as a concept, say D. To denote the relationship between D and R, we denote as *Range*(R, D). It is obvious that C, D are both subsets of $\Delta^{\mathcal{I}}$.

Domain and range constraint are widely supported by modern KB description languages such as OWL. With the presence of domain and range axioms, reasoning tasks relating to role restrictions can be interpreted to: if $R(x, y)$, *Domain*(R, C) and *Range*(R, D), then $x \in C$ and $y \in D$.

The above mentioned idea raises a new kind of optimization technique which we call *Role Absorption*[3] [5] [14]. In many description languages, especially some old KB systems, domain and range axioms are not directly supported, but can trivially transformed into GCIs. That is, restricting the domain of a role R to be a concept C is equivalent to adding an axiom asserting that the concept whose instances are related to some other individuals by role R is subsumed by C (i.e. $\exists R.\top \sqsubseteq C$); similarly, restricting the range of a role R to be concept D is equivalent to adding an axiom

[3] In some papers, it is also called *Domain and Range absorption*

asserting that the most general concept is subsumed by the concept whose instances are related by role $\mathcal{R}$ only to instances of D (i.e. $\top \sqsubseteq \forall R.D$) [10]. The problem of such a transformation is that these GCIs are not suitable to be converted to trigger rules (absorption). As we discussed above, the complexity of reasoning for trigger rules and GCIs is very different. Most modern reasoners, such as RACER, FaCT++ or Pellet, act much less well with KBs containing a significant number of unabsorbable GCIs. Unfortunately, many modern knowledge bases contain large numbers of different roles — each with domain and role constrains — thus the resulting KBs contain many unabsorbable GCIs. To solve this problem, we have the following proposition:

Proposition 4.4 *Let $\mathcal{T}$ be an arbitrary TBox, $\mathcal{R}$ be a role*

1. An interpretation $\mathcal{I}$ satisfies $\mathcal{T}$ and an axiom $\exists R.\top \sqsubseteq C$ iff $\mathcal{I}$ satisfies $\mathcal{T} \cup \{Domain(R, C)\}$.

2. An interpretation $\mathcal{I}$ satisfies $\mathcal{T}$ and $\top \sqsubseteq \forall R.D$ iff $\mathcal{I}$ satisfies $\mathcal{T} \cup \{Range(R, D)\}$.

Proof

The proof for the above proposition is obvious considering the definition of domain and range. For the first claim, we consider the iff direction first. $\mathcal{I}$ satisfies $\mathcal{T}$ implies that $\mathcal{I}$ satisfies every axiom of $\mathcal{T}$. To prove this assertion, we only need to prove that if $\mathcal{I}$ satisfies $\exists R.\top \sqsubseteq C$, $\mathcal{I}$ also satisfies $Domain(\mathcal{R}, C)$. Suppose that there is some $\langle x, y\rangle \in \mathcal{R}^{\mathcal{I}}$, and $x \notin C^{\mathcal{I}}$. However, $\langle x, y\rangle \in \mathcal{R}^{\mathcal{I}}$ implies $x \in (\exists \mathcal{R}.\top)^{\mathcal{I}}$. Thus, $x \in \mathcal{C}^{\mathcal{I}}$ since $(\exists \mathcal{R}.\top)^{\mathcal{I}}$ is a subset of $\mathcal{C}^{\mathcal{I}}$ — a contradiction. Similarly, it is not difficult to prove the second part of this proposition.

This kind of rewriting technique can be extended to deal with a wider range of axioms. First let us prove an equivalence: $(\exists R.C \equiv \exists R.C \sqcap \exists R.\top)$. It is obvious that $(\exists R.C \sqcap \exists R.\top \sqsubseteq \exists R.C)$. We need to prove $(\exists R.C \sqsubseteq \exists R.C \sqcap \exists R.\top)$. Let us check

the satisfiability of $(\exists R.C \sqcap \neg(\exists R.C \sqcap \exists R.\top))$. This concept can be simplified as $(\exists R.C \sqcap \forall R.\bot)$. It is obvious that this concept is unsatisfiable. Therefore, $(\exists R.C \equiv \exists R.C \sqcap \exists R.\top)$ holds. From this equivalence, we can rewrite the axiom such as $(\exists R.C \sqsubseteq D)$ to $(\exists R.C \sqcap \exists R.\top \sqsubseteq D)$, i.e., $(\exists R.\top \sqsubseteq (\forall R.\neg C) \sqcup D)$. Thus, the axiom in the form of $(\exists R.C \sqsubseteq D)$ can be absorbed as $Domain(R, D \sqcup \forall R.\neg C)$. Similarly, the axiom in the form of $(D \sqsubseteq \forall R.C)$ can be rewritten as $(\exists R.\neg C \sqsubseteq \neg D)$. It can also be absorbed as $Domain(R, \neg D \sqcup \forall R.C)$.

4.4 Complexity of Absorption

4.4.1 Ambiguity of Equivalence

In the preceding sections, we have mentioned several times the "left-hand side" when describing the absorption algorithms. Please note whenever we mentioned the left-hand side, we referred to the left-hand side of the symbol "⇒". If we apply absorption algorithms without considering this restriction, it may generate an invalid absorption during unfolding.

Consider the following example: a TBox $\mathcal{T}$ contains two parts, $\mathcal{T}_u$ and $\mathcal{T}_g$. $\mathcal{T}_u = \{A \equiv B\}$[4] and $\mathcal{T}_g = \{\top \sqsubseteq \neg B \sqcup C\}$. A and B are atomic concepts; C is an arbitrary concept. By applying **Proposition 4.3**, one may consider that B does not occur on the "left-hand side" of $\mathcal{T}_u$ and get the following absorbed TBox $\mathcal{T}'$:

$$\mathcal{T}_u^{'} = \{A \equiv B; B \Rightarrow C\}$$
$$\mathcal{T}_g^{'} = \phi$$

In fact, the above absorption is invalid. One example is that concept $(B \sqcap \neg A)$ is now.

We can do reasoning by applying lazy unfolding and semantic tableau as the following:

[4]In some papers and reasoners, the symbol "≡" is used as a two way reasoning symbol in $\mathcal{T}_u$ to represent an concept definition axiom.

(1) Add B and $\neg A$ to $\mathcal{L}(x)$ because the expression is a conjunction.

(2) The axiom $A \equiv B$ is the same as $B \equiv A$ because the left-hand side and the right-hand side of "$\equiv$" is exchangeable. Therefore, from $B \equiv A$, we are able to add A to $\mathcal{L}(x)$ by lazy unfolding. This causes conflict. Thus, the concept $(B \sqcap \neg A)$ with respect to $\mathcal{T}$ is unsatisfiable.

However, the reasoning procedure can also be applied as follows:

(1) Add B and $\neg A$ to $\mathcal{L}(x)$ because the expression is a conjunction.

(2) Considering the axiom $B \Rightarrow C$, add concept C to $\mathcal{L}(x)$ by applying lazy unfolding.

Then the $\mathcal{L}(x)$ can not be further unfolded, and no conflict occurs. Thus, we claim that the concept $(B \sqcap \neg A)$ with respect to $\mathcal{T}$ is satisfiable!

To avoid such kind of invalid reasoning, many reasoners restrict the kind of absorptions mentioned above. Because of this restriction, some axioms may not be absorbed successfully. As a result, reasoning performance for such kind of KBs could be affected significantly.

In fact, the main reason of the above mentioned problem is caused by one-way reasoning mechanism of *lazy unfolding* and the required two way reasoning for the symbol "$\equiv$" in an axiom. We can easily solve this problem by rewriting the axiom $\{A \equiv B\}$ into two rule axioms $\{A \Rightarrow B; \neg A \Rightarrow \neg B\}$ instead. Of course, we have other ways to rewrite this axiom. For example, we can also rewrite it to $\{B \Rightarrow A; \neg B \Rightarrow \neg A\}$ as well. We can also easily prove that the above pairs are also valid absorptions of the axiom $A \equiv B$. Therefore, in this thesis we do not use the symbol "$\equiv$" but use only the symbol "$\Rightarrow$" to represent axioms in $\mathcal{T}_u$ to emphasize one-way reasoning.

Moreover, there are also other ways to solve this kind of problem. For example, we can redefine the rule for lazy unfolding as: whenever the symbol in $\mathcal{T}_u$ is "$\equiv$" for an atomic concept A, we replace both A and $\neg A$ in $\mathcal{L}(x)$ by their definitions. In the above example, we replace A by B and $\neg A$ by $\neg B$ whenever we have A or $\neg A$ in

$\mathcal{L}(x)$.

4.4.2 Non-determinism of Absorption Results

By applying trigger rules, i.e., axioms with "$\Rightarrow$", instead of axioms with "$\equiv$" in a TBox unfoldable part, we are able to eliminate above-mentioned problem with absorption. However, the non-determinism of absorption results still makes absorption effectiveness very unpredictable. We say the non-determinism of absorption means the result of absorption is undeterminable. That is, we may have several choices to rewrite $\mathcal{T}_g$ to $\mathcal{T}_u$.

The reason of absorption non-determinism is obvious. For example, to rewrite $\mathcal{T}_g = \{\neg A \sqcup B\}$ (A and B are atomic concepts) to $\mathcal{T}_u$, we have two choices: either absorb it into $\{A \Rightarrow B\}$ or $\{B \Rightarrow \neg A\}$. The more complex a TBox is, the more possible absorption results we may have. Please see the following example.

Let $\mathcal{T}$ be a TBox, $\mathcal{T} = \mathcal{T}_u \sqcup \mathcal{T}_g$, and $\mathcal{T}_u = \phi$, $\mathcal{T}_g = \{\neg A \sqcup B; A \sqcup C\}$; A, B and C are all atomic concepts.

By applying the absorption rules described in previous sections, we can easily see that the following absorption results are all valid absorptions:

(1) $\mathcal{T}_1 = \mathcal{T}_{u1} \sqcup \mathcal{T}_{g1}$:
$\mathcal{T}_{u1} = \{A \Rightarrow B; \neg A \Rightarrow C; \neg B \Rightarrow C\}$; $\mathcal{T}_{g1} = \phi$

(2) $\mathcal{T}_2 = \mathcal{T}_{u2} \sqcup \mathcal{T}_{g2}$:
$\mathcal{T}_{u2} = \{A \Rightarrow B; \neg A \Rightarrow C; \neg C \Rightarrow B\}$; $\mathcal{T}_{g2} = \phi$

(3) $\mathcal{T}_3 = \mathcal{T}_{u3} \sqcup \mathcal{T}_{g3}$:
$\mathcal{T}_{u3} = \{\neg B \Rightarrow \neg A; \neg A \Rightarrow C\}$; $\mathcal{T}_{g3} = \phi$

(4) $\mathcal{T}_4 = \mathcal{T}_{u4} \sqcup \mathcal{T}_{g4}$:

$\mathcal{T}_{u4} = \{\neg B \Rightarrow \neg A; \neg C \Rightarrow A\}$; $\mathcal{T}_{g4} = \phi$

(5) $\mathcal{T}_5 = \mathcal{T}_{u5} \sqcup \mathcal{T}_{g5}$:
$\mathcal{T}_{u5} = \{A \Rightarrow B; \neg C \Rightarrow A\}$; $\mathcal{T}_{g5} = \phi$

Any of the above absorption results completely absorbed $\mathcal{T}_g$ to $\mathcal{T}_u$. The many possibilities of absorption results may increase in the presence of *roles* since we may have more options to absorb an axiom by applying either domain and range absorption or concept absorption. For example, the axiom $\{A \sqcap \exists R.B \sqsubseteq C\}$ can be rewritten to $\{\exists R.B \sqsubseteq C \sqcup \neg A\}$. This axiom can be absorbed by using extended role absorption to $Domain(R, \forall R.\neg B \sqcup C \sqcup \neg A)$. Obviously, this axiom can also be rewritten as $A \sqsubseteq \forall R.(\neg B \sqcup C)$. Thus, we can absorb it as the axiom $\{A \Rightarrow \forall R.(\neg B \sqcup C)\}$.

Most modern knowledge bases are very complex. For example, the GALEN KB has thousands of atomic concepts, while the TAMBIS KB has hundreds of atomic roles. Among so many absorption choices, which one is the best? How to find the "best" absorption for a specific KB? These two questions are the main topic of the following sections.

4.4.3 Absorption Cycles

In **Lemma 4.3** and **Lemma 4.4**, to absorb an axiom from $\mathcal{T}_g$, sometimes we also need to push back another axiom into $\mathcal{T}_g$. The action of "push back" may cause absorption cycles. For example, the following is a TBox $\mathcal{T} = \mathcal{T}_u \sqcup \mathcal{T}_g$:

$\mathcal{T}_u = \{A \Rightarrow B; B \Rightarrow A;\}$
$\mathcal{T}_g = \{\top \sqsubseteq B \sqcup D\}$

By applying **Lemma 4.3** to absorb $\mathcal{T}_g$, the first step is to add $\neg B \Rightarrow D$ to $\mathcal{T}_u$ and put $\top \sqsubseteq D \sqcup A$ back to $\mathcal{T}_g$. Then absorb $\neg A \Rightarrow D$ to $\mathcal{T}_u$ and put back $\top \sqsubseteq B \sqcup D$ to $\mathcal{T}_g$. Thus, a cycle occurs.

To solve this problem, a *fixpoint* or block needs to be considered. For example, because the axiom $\top \sqsubseteq B \sqcup D$ that needs to be put back to $\mathcal{T}_g$ is not a new axiom, we can just discard it. Therefore, the absorption result of $\mathcal{T}$ is as follows:

$$\mathcal{T}_u = \{A \Rightarrow B; B \Rightarrow A; \neg B \Rightarrow D; \neg A \Rightarrow D\}$$
$$\mathcal{T}_g = \phi$$

4.5 Absorption Algorithms

4.5.1 Standard Absorption Algorithm

In previous sections, we have discussed absorption and its complexity. In this section, let us first take a look at the absorption algorithms already widely applied in many reasoners. We refer to such kind of absorption algorithm as *standard absorption algorithm*. Axiom absorption in a standard absorption algorithm relies on the form of an axiom. For example, if an axiom has the form $(A \sqsubseteq C)$ in $\mathcal{T}_g$ where $A \in NC$. If neither A nor $\neg A$ has a rule definition in $\mathcal{T}_u$, then this axiom can be easily absorbed into $\mathcal{T}_u$ as $(A \Rightarrow C)$. Based on this idea, the formulas described in Figure 10 are widely employed in standard absorption algorithm to absorb GCI from $\mathcal{T}_g$ to $\mathcal{T}_u$.

$$\begin{array}{lcl} \{C_1 \sqcap C_2 \sqsubseteq D\} & \Longleftrightarrow & \{C_1 \Rightarrow \neg C_2 \sqcup D\} \text{ or } \{C_2 \Rightarrow \neg C_1 \sqcup D\} \\ \{C \sqsubseteq D_1, C \sqsubseteq D_2\} & \Longleftrightarrow & \{C \Rightarrow D_1 \sqcap D_2\} \\ \{C \equiv D\} & \Longleftrightarrow & \{C \Rightarrow D, D \Rightarrow C\} \end{array}$$

Figure 10: Axiom equivalence used in absorption

The standard absorption algorithm can be summarized as follows (see [9] for more details).

Given a KB divided into an unfoldable part $\mathcal{T}_u$ and a general part $\mathcal{T}_g$. If an axiom has the format $A \sqsubseteq C$ or $A \equiv C$ in $\mathcal{T}$ where $A \in NC$ and neither A nor $\neg A$ occur on the left-hand side of $\mathcal{T}_u$, then put this axiom into $\mathcal{T}_u$. Otherwise, put this axiom into $\mathcal{T}_g$. Then, initialize a set $\mathcal{T}_g'$ to be empty, and convert any axioms of the form

$\{(C \equiv D) \in \mathcal{T}_g\}$ *into an equivalent pair of axioms* $C \sqsubseteq D$ *and* $\neg C \sqsubseteq \neg D$ *first. Then, apply the following steps to all axioms in* $\mathcal{T}_g$ *in the form of* $C \sqsubseteq D$*:*

For each $C \sqsubseteq D \in \mathcal{T}_g$*,*

(i) Initialize a set $\boldsymbol{G} = \{\neg D, C\}$*, representing the axiom in the form* $\top \sqsubseteq \neg \sqcap \{\neg D, C\}$ *(i.e.* $\top \sqsubseteq D \sqcup \neg C$*).*

(ii) If for some $A \in \mathbf{G}$*, there is a rule definition* $(A \Rightarrow C) \in \mathcal{T}_u$*, then absorb the general axiom into a primitive definition axiom so that it becomes*

$$A \Rightarrow \sqcap\{C, \neg \sqcap (\mathbf{G} \setminus \{A\})\},$$

and exit.

(iii) If for some $A \in \mathbf{G}$*, there is no rule definition in* $\mathcal{T}_u$*, then absorb the general axiom into the primitive definition axiom so that it becomes*

$$A \Rightarrow \neg \sqcap (\mathbf{G} \setminus \{A\}),$$

and exit.

(iv) If for some $A \in \mathbf{G}$*, there is an axiom* $(A \equiv D) \in \mathcal{T}_u$*, then substitute* $A \in \mathbf{G}$ *with* D

$$\mathbf{G} \longrightarrow \{D\} \sqcup \mathbf{G} \setminus \{A\}$$

and return to step (ii)*.*

(v) If for some $\neg A \in \mathbf{G}$ *there is an axiom* $(A \equiv D) \in \mathcal{T}_u$*, then substitute* $\neg A \in \mathbf{G}$ *with* $\neg D$

$$\mathbf{G} \longrightarrow \{\neg D\} \sqcup \mathbf{G} \setminus \{\neg A\}$$

and return to step (ii)*.*

(vi) If the axiom can not be absorbed, add this axiom to $\mathcal{T}_g^{'}$ *and exit.*

Upon experimental results, the above mentioned absorption algorithm has a significant gain in optimizing the reasoning performance compared to reasoning without

absorption. However, in some cases the absorption algorithm itself may be also time-consuming. For example, in step (iv) and step (v) we are substituting an atomic concept by its definition. For a very complex KB, the substitution itself may take some time. To solve this problem, in RACER, an enhanced absorption algorithm is employed. That is, a "split" algorithm is applied instead of a substitution if there are axioms left in $\mathcal{T}_g$ containing the atomic concept which has a complete definition in $\mathcal{T}_u$. For example, a concept definition axiom $A \equiv C$ can be split into two axioms $\{A \Rightarrow C\}$ and $\{\top \sqsubseteq A \sqcup \neg C\}$, the first axiom can be used to absorb any general axiom containing $\neg A$; the second axiom can be put back into $\mathcal{T}_g$ for further absorption.

4.5.2 Criteria of the "Best" Absorption

It is obvious that both the standard absorption algorithm mentioned above and the enhanced absorption algorithm employed by RACER are non-deterministic. In the first place, there may exist multiple choices to divide $\mathcal{T}$ into unfoldable and general parts. For example, if $\mathcal{T}$ contains more than one complete definition for a concept name A, then one of them must be selected as a definition in $\mathcal{T}_u$ while the others are treated as GCIs in $\mathcal{T}_g$. The criteria of which one should be selected has never been specified. Moreover, the intuition behind the non-deterministic of the standard absorption algorithm is that the target GCI may contain multiple atomic concepts into which this axiom could be absorbed. For example, for a GCI: $\{\top \sqsubseteq \neg A_1 \sqcup \neg A_2\}$ in $\mathcal{T}_g$, and two axioms $\{A_1 \Rightarrow C\}$ and $\{A_2 \Rightarrow D\}$ in $\mathcal{T}_u$, then the axiom could be absorbed into either A_1 to give $A_1 \Rightarrow C \sqcap \neg A_2$ or into A_2 to give $A_2 \Rightarrow D \sqcap \neg A_1$. In short, the result of the standard absorption algorithms depends on the axiom order and axiom format in the target KB upon the steps specified by the algorithm. Moreover, there is an implicit restriction in the above mentioned algorithms which prevents both A and $\neg A$ from being absorbed into $\mathcal{T}_u$.

Now, for a specific KB, by following the absorption formula we may have many

absorption results. Since the basic motivation is to improve reasoning performance for a specific KB, then among all the possible absorption possibilities, which one is the best?

To answer this question, we have to go back to the underlying idea of absorption and tableau algorithms. As we know, the root cause of the inefficiency of tableau algorithms is mainly caused by the disjunctions in the specific KB; while the efficiency of absorption is that it "absorbs" axioms from $\mathcal{T}_g$ to $\mathcal{T}_u$ which avoids the unnecessary unfolding and decreases the number of disjunctions in $\mathcal{T}_g$. Therefore, based on the above analysis, we propose a criterion for the "best" absorption is: **the best absorption is the one which contains the least number of disjunctions in $\mathcal{T}_g$**[5]. Based on this criterion, in the following sections, we discuss some other absorption algorithms which will improve reasoning performance.

4.5.3 Heuristic Absorption Algorithm

As we have discussed above, the standard absorption algorithm depends on the axiom order and axiom format in the target KB, while the content of a KB is irrelevant to the KB format. Thus, we can not expect that the standard absorption algorithm generates the "best" one, i.e., it contains the least number of disjunctions in $\mathcal{T}_g$. Therefore, an intuition is that the "best" absorption algorithm has to be independent of the format of the KB. To develop an absorption algorithm irrespective to the format of a TBox, the intuition is that we should evaluate an axiom absorption by considering the entire TBox instead of one specific axiom. In addition, a possible good way to find the "best" absorption is to heuristically absorb as many general axioms as possible into $\mathcal{T}_u$. To simplify our discussion, the objective of heuristic absorption algorithms that we will discuss in the following sections is to reduce the number of axioms instead of the number of disjunctions in T_g. We leave the topic of heuristic

[5]Upon experimental result, the criterion we propose here may not always be the "best" absorption regarding to reasoning. Please refer to Chapter 7.

absorption algorithms purely based on the number of disjunctions in T_g to our future work (see **Chapter 7**).

The first step of the heuristic absorption algorithm is normalization and simplification as discussed in **Chapter 3**. In this step, we first convert all normalized axioms from a TBox $\mathcal{T}$ into $\mathcal{T}_g$, and $\mathcal{T}_u$ is initialized to be empty. We do not divide a TBox in the first step because in some KBs, even though the axiom can be put in $\mathcal{T}_u$ by using the standard absorption algorithm, the axiom may not be in fact an "unfoldable" one (see **Section 4.5.1**). For example, assume a TBox $\mathcal{T}$=$\{A_0 \equiv B_0 \sqcap C_0 \sqcap D_0, B_0 \equiv A_0 \sqcap E_0 \sqcap F_0, \top \sqsubseteq B_0 \sqcup \exists R_1.C_1\}$ (suppose $A_0, B_0, C_0, D_0, E_0, F_0$ are all atomic concepts). The first two axioms would be directly put in T_u by standard absorption algorithms. Thus, this TBox becomes $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$, $\mathcal{T}_u = \{A_0 \equiv B_0 \sqcap C_0 \sqcap D_0, B_0 \equiv A_0 \sqcap E_0 \sqcap F_0\}$; $\mathcal{T}_g = \{\top \sqsubseteq B_0 \sqcup \exists R_1.C_1\}$[6]. The only possibility to absorb the third axiom is to absorb it into B_0. Unfortunately, B_0 can not be directly absorbed based on the standard absorption algorithm since B_0 already has a complete definition in $\mathcal{T}_u$. However, if we simplify and normalize the entire TBox first, then apply absorption algorithm to the entire TBox, we may be able to absorb all axioms from $\mathcal{T}_g$ even by following the standard absorption algorithms.

First, rewrite the TBox to the following by simplification and normalization:

$\mathcal{T}_u \equiv \phi$

$\mathcal{T}_g$ contains the following axioms:

(1)$\{\neg A_0 \sqcup B_0\}$;

(2)$\{\neg B_0 \sqcup A_0\}$;

(3)$\{C_0 \sqcup \neg A_0\}$;

(4)$\{D_0 \sqcup \neg A_0\}$;

(5)$\{E_0 \sqcup \neg B_0\}$;

(6)$\{F_0 \sqcup \neg B_0\}$;

[6]We keep the symbol "$\equiv$" for the standard absorption algorithms in $\mathcal{T}_u$ since it is still being used in other papers.

(7)$\{B_0 \sqcup \exists R_1.C_1\}$;

From axioms (1) and (2), we conclude that concept $A_0 \equiv B_0$. This axiom can be added into the unfoldable part $\mathcal{T}_u$, and the axioms (1) and (2) should be removed from $\mathcal{T}_g$. At the same time, we are able to replace all occurances of A_0 in $\mathcal{T}_g$ by B_0. Now the TBox $\mathcal{T}$ becomes[7]:

$\mathcal{T}_u$:

(1)$\{A_0 \Rightarrow B_0\}$;

(2)$\{\neg A_0 \Rightarrow \neg B_0\}$;

$\mathcal{T}_g$:

(1)$\{C_0 \sqcup \neg B_0\}$;

(2)$\{D_0 \sqcup \neg B_0\}$;

(3)$\{E_0 \sqcup \neg B_0\}$;

(4)$\{F_0 \sqcup \neg B_0\}$;

(5)$\{B_0 \sqcup \exists R_1.C_1\}$;

Now this TBox $\mathcal{T}$ can be completely absorbed as the following:

$\mathcal{T}_u$:

(1)$\{A_0 \Rightarrow B_0\}$;

(2)$\{\neg A_0 \Rightarrow \neg B_0\}$;

(3)$\{\neg C_0 \Rightarrow \neg B_0\}$;

(4)$\{\neg D_0 \Rightarrow \neg B_0\}$;

(5)$\{\neg E_0 \Rightarrow \neg B_0\}$;

(6)$\{\neg F_0 \Rightarrow \neg B_0\}$;

(7)$\{\neg B_0 \Rightarrow \exists R_1.C_1\}$;

$\mathcal{T}_g = \phi$:

[7]We do not use the symbol "$\equiv$" in $\mathcal{T}_u$ in the following discussions to avoid ambiguity.

If we analyze the underlying idea for absorption in detail, we can find that there are two directions to absorb more axioms from $\mathcal{T}_g$ to $\mathcal{T}_u$: one direction is to absorb as many axioms as possible in $\mathcal{T}_g$; the other direction is to reduce the number of axioms in $\mathcal{T}_u$ as much as possible. The fundamental idea of the latter direction is based on the absorption restrictions: the maximum number of axioms in $\mathcal{T}_u$ depends on total number of atomic concepts in the target TBox if we do not consider the role absorption. Therefore, if we reduce the number of concept names occurring on the left-hand side of axioms in $\mathcal{T}_u$, i.e., the number of axioms in $\mathcal{T}_u$, we may be able to absorb more axioms from $\mathcal{T}_g$. Eventually, we may be able to reduce the number of disjunctions in $\mathcal{T}_g$ to improve the reasoning performance. From this point of view, let us discuss the algorithms for this direction first. Please refer to the following example:

A given TBox $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$ is given as follows:

$\mathcal{T}_u$ contains:

$(1)(A \Rightarrow B)$

$(2)(C \Rightarrow B)$

$(3)(B \Rightarrow (A \sqcup C))$

$\mathcal{T}_g$ contains:

$(1)(\top \sqsubseteq A \sqcup C \sqcup \exists R.D)$

It is obvious that the axiom in $\mathcal{T}_g$ can not be further absorbed by the standard absorption algorithms because all of A, B, C already occur on the left-hand side of $\mathcal{T}_u$. However, if we analyze the axioms (1) and (2), we can rewrite them to one axiom such as $\{A \sqcup C \Rightarrow B\}$. Then let us consider axiom (3), and we can rewrite axiom (1), (2), (3) to another axiom $\{B \equiv (A \sqcup C)\}$ or a pair of axioms $\{B \Rightarrow (A \sqcup C)\}$ and $\{\neg B \Rightarrow (\neg A \sqcap \neg C)\}$. Now the number of axioms in $\mathcal{T}_u$ can be reduced from three to two. Therefore, axiom (1) in $\mathcal{T}_g$ can be easily absorbed into $\mathcal{T}_u$ as $\{\neg A \Rightarrow C \sqcup \exists R.D\}$ since neither A nor $\neg A$ occur on the left-hand side in $\mathcal{T}_u$ any more. Now the TBox $\mathcal{T}$ can be completely absorbed. We refer to this type of absorption as *primitive definition*

to complete definition conversion.

To be more precise, the absorption algorithm procedure of *primitive definition to complete definition conversion* can be summarized by the following steps.

Given an arbitrary TBox $\mathcal{T}$. Suppose A is an atomic concept.

1. *Simplify and normalize* $\mathcal{T}$. *After normalization,* $\mathcal{T}_u = \emptyset$ *and* $\mathcal{T}_g$ *contains a set of axioms in the form* $\top \sqsubseteq C \sqcup D$ *where* C *and* D *are either atomic concepts or modal expressions. Then each axiom in* $\mathcal{T}_g$ *can be expressed as a set. For example, the set* $\boldsymbol{G} = \{C, D\}$ *represents the axiom in the form* $\top \sqsubseteq C \sqcup D$. *As a consequence,* $\neg\boldsymbol{G}$ *represents a set containing the concept* $\neg C \sqcap \neg D$. *Therefore, in each axiom of* $\mathcal{T}_g$, *it only contains* A, *or* $\neg A$, *or neither. We also need a function* con() *which returns for a given set* $\boldsymbol{G}$ *its represented concept, e.g., if* $\boldsymbol{G}$ $= \{C, D\}$, *then* $\mathrm{con}(\boldsymbol{G})$ *returns* $C \sqcup D$.

2. *Initialize two sets* $\mathcal{T}_{g_1}$, $\mathcal{T}_{g_2}$ *to be empty and consider* A *the chosen (fixed) atomic concept.*

3. *For any set* $\boldsymbol{G}$, *if* $A \in \boldsymbol{G}$, *then remove* $\boldsymbol{G}$ *from* $\mathcal{T}_g$ *and add an element* $\neg(\boldsymbol{G}\backslash\{A\})$ *to* $\mathcal{T}_{g_1}$; *if* $\neg A \in \boldsymbol{G}$, *then remove* $\boldsymbol{G}$ *from* $\mathcal{T}_g$ *and add an element* $\boldsymbol{G}\backslash\{\neg A\}$ *to* $\mathcal{T}_{g_2}$; *otherwise, keep* $\boldsymbol{G}$ *in* $\mathcal{T}_g$.

4. *For each element* $\boldsymbol{G}_2$ *in* $\mathcal{T}_{g_2}$, *if* $\boldsymbol{G}_2$ *also appears in* $\mathcal{T}_{g_1}$,

   (a) *remove* $\boldsymbol{G}_2$ *from both* $\mathcal{T}_{g_1}$ *and* $\mathcal{T}_{g_2}$;

   (b) *add the axiom* $\{A \Rightarrow \mathrm{con}(\mathbf{G_2})\}$ *and* $\{\neg A \Rightarrow \mathrm{con}(\neg\mathbf{G_2})\}$ *to* $\mathcal{T}_u$.

5. *For each element set* $\boldsymbol{G}'_1$ *left in* $\mathcal{T}_{g_1}$, *create a new set* $\neg\boldsymbol{G}'_1 \cup \{A\}$ *and put it back into* $\mathcal{T}_g$; *for each element set* $\boldsymbol{G}'_2$ *in* $\mathcal{T}_{g_2}$, *create a new set* $\boldsymbol{G}'_2 \cup \{\neg A\}$ *and put it back into* $\mathcal{T}_g$.

The above algorithm can be further improved in step (4) by checking more than one element sets in G_2. The gain is obvious if many axioms can be absorbed into

one concept definition. However, in the worst case, if the algorithm fails to find a complete definition for a specified atomic concept, the performance may not be improved significantly.

Now let us consider how to absorb axioms from $\mathcal{T}_g$ in a more general way. As we have discussed in previous sections, there usually have more than one choice for absorbing a specific axiom from $\mathcal{T}_g$ to $\mathcal{T}_u$. For example, in the following TBox $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$, suppose A, B, C, D, E are all atomic concepts:

$\mathcal{T}_u = \phi$

$\mathcal{T}_g$ contains the following axioms:

(1) $\top \sqsubseteq (\neg A \sqcup C)$;

(2) $\top \sqsubseteq (B \sqcup D)$

(3) $\top \sqsubseteq (\neg B \sqcup \neg C)$

(4) $\top \sqsubseteq (\neg A \sqcup E)$

To absorb axiom (1), we have at least two choices: absorb it into $A \Rightarrow C$ or $\neg C \Rightarrow \neg A$. The intuitive question is: which one is better? One intuitive response is that we should choose the one which absorbs more GCIs, i.e., more disjunctions from $\mathcal{T}_g$ upon the criterion we set in the previous section[8]. In this case, it is obvious that absorbing the axiom into $A \Rightarrow C$ is better than absorbing it into $\neg C \Rightarrow \neg A$ because in this TBox, there are two axioms contain $\neg A$ but only one axiom contains C. If we absorb axioms based on $\neg A$, we can reduce two axioms from $\mathcal{T}_g$. Thus, *the concept occurrence frequency in $\mathcal{T}_g$ is an indicator to tell how many axioms could be absorbed.* However, an algorithm simply based on atomic concept occurrence frequency in $\mathcal{T}_g$ may not always be the best choice. Please consider the following example TBox $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$:

$\mathcal{T}_u$ contains the following axioms:

(1) $B \Rightarrow A$;

[8]To be more precise, the number of axioms may not necessarily be directly proportional to the number of disjunctions in $\mathcal{T}_g$. We leave the discussion about this topic to our future work.

(2) $C \Rightarrow A$

$\mathcal{T}_g$ contains the following axioms:

(1) $\top \sqsubseteq (\neg A \sqcup C)$;

(2) $\top \sqsubseteq (\neg B \sqcup \neg C)$

(3) $\top \sqsubseteq (\neg A \sqcup E)$

In the above example, if we use the same criterion as we used in the above example to absorb axiom (1) and (3) at the same time because there are two axioms contain $\neg A$, we may find out that axiom (2) can not be absorbed. To absorb as many axioms in $\mathcal{T}_g$ as possible, we developed the following heuristic absorption algorithm as an extension to the above mentioned *primitive definition to complete definition conversion*. To simplify our following discussions, we refer to this absorption algorithm as *basic absorption algorithm*.

Given an arbitrary TBox $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$. $\mathcal{T}_u$ contains a set of axioms in the form of $A \Rightarrow C$. $\mathcal{T}_g$ contains a set of axioms in the form of $\top \sqsubseteq D$.

1. *Simplify and normalize $\mathcal{T}$ as described above.*

2. *Suppose c_p is the total number of appearance of A in $\mathcal{T}_g$; and c_n is the total number of appearance of $\neg A$ in $\mathcal{T}_g$. Among all atomic concepts $A(c_p, c_n)$, ignore the concepts where both c_p and c_n equal to zero. Then divide these atomic concepts into two groups: the concepts in the first group where one of c_p or c_n is 0; put the remaining atomic concepts in the other group.*

3. *Select the atomic concept from the first group which has the greatest value among all the $c_p's$ and $c_n's$ and absorb all the axioms in $\mathcal{T}_g$ which contain this atomic concept*[9] *to $\mathcal{T}_u$.*

4. *Repeat the steps 2 to 4 until there is no item left in the first group.*

[9] If the greatest value is c_p, then the selected concept is A. Otherwise, the selected concept is $\neg A$.

5. *Select the atomic concept from the second group with the greatest value among all the $c_p's$ and $c_n's$, then absorb all the axioms in $\mathcal{T}_g$ which contain this atomic concept*[10] *to $\mathcal{T}_u$ if these axioms can be absorbed.*

6. *Repeat the steps 2 to 6 until there is no item in the second group or no more concepts can be absorbed.*

Theoretically, the above mentioned heuristic absorption algorithm is based on the entire TBox, and in each absorption step, it tries to absorb as many axioms as possible. Therefore, compared to the standard absorption algorithm, it is supposed to be better. However, similar to the standard absorption, there still exists the implicit restriction that A and $\neg A$ are not allowed to occur at the same time on the left-hand side of $\mathcal{T}_u$. This kind of restriction limits the maximum number of axioms in $\mathcal{T}_u$ to be the number of atomic concepts in the TBox. In fact, this limitation can be relaxed because of **Lemma 4.3** and **Lemma 4.4**. Thus, we can develop an extended absorption algorithm in addition to the basic heuristic absorption algorithm.

4.5.4 Extended Heuristic Absorption Algorithm

Based on **Lemma 4.3** and **Lemma 4.4**, we are able to develop an absorption algorithm by allowing both positive and negative atomic concept names to occur on the left-hand side of $\mathcal{T}_u$.

Consider the following example ($A, B, C \in NC$):

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;

$\mathcal{T}_u = \phi$;

$\mathcal{T}_g$ contains the axioms:

$\top \sqsubseteq \neg A \sqcup \neg B \sqcup \neg C$,

$\top \sqsubseteq A \sqcup B \sqcup C$,

$\top \sqsubseteq B \sqcup \exists R_1.C_1$,

[10] Same as in the step 3.

$\top \sqsubseteq C \sqcup \exists R_2.C_2$,

$\top \sqsubseteq A \sqcup \forall R_3.C_3$

Let us apply the *basic heuristic absorption algorithm* directly since no obvious complete atomic concept definition can be found.

The statistical values can be calculated as the following,

$A(2,1)$, $B(2,1)$, $C(2,1)$, $C_1(0,0)$, $C_2(0,0)$

Suppose we absorb axioms containing A first since no atomic concepts falls in the first group for the *basic heuristic absorption algorithm*. The TBox is as follows after the absorption,

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;

$\mathcal{T}_u$ contains:

$\neg A \Rightarrow (B \sqcup C) \sqcap \forall R_3.C_3$;

$\mathcal{T}_g$ contains:

$\top \sqsubseteq \neg A \sqcup \neg B \sqcup \neg C$,

$\top \sqsubseteq B \sqcup \exists R_1.C_1$,

$\top \sqsubseteq C \sqcup \exists R_2.C_2$

The statistical values are the following[11] for the above TBox,

$A(0,1)$, $B(1,1)$, $C(1,1)$

Since $\neg A$ already has a rule axiom, the axiom containing $\neg A$ can not be absorbed. Therefore, we check the atomic concepts in the second group. Suppose we absorb axioms containing $\neg B$ in $\mathcal{T}_g$ this time, the TBox is as follows after the absorption,

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;

$\mathcal{T}_u$ contains:

$\neg A \Rightarrow (B \sqcup C) \sqcap \forall R_3.C_3$,

$B \Rightarrow \neg A \sqcup \neg C$;

$\mathcal{T}_g$ contains:

[11] We ignore the concepts where both c_p and c_n equal to zero.

$\top \sqsubseteq B \sqcup \exists R_1.C_1$

$\top \sqsubseteq C \sqcup \exists R_2.C_2$

The statistical values are as the following for the above TBox,

$B(1,0)$, $C(1,0)$.

Eventually, this TBox can be absorbed as follows by applying the *basic heuristic absorption algorithm*,

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;

$\mathcal{T}_u$ contains:

$\neg A \Rightarrow (B \sqcup C) \sqcap \forall R_3.C_3$,

$B \Rightarrow \neg A \sqcup \neg C$,

$\neg C \Rightarrow \exists R_2.C_2$;

$\mathcal{T}_g$ contains:

$\top \sqsubseteq B \sqcup \exists R_1.C_1$

There is still one axiom in $\mathcal{T}_g$ that can not be absorbed . If we apply **Lemma 4.3** and **Lemma 4.4**, the above absorption can be further absorbed as the following,

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;

$\mathcal{T}_u$ contains the axioms:

(1)$\neg A \Rightarrow (B \sqcup C) \sqcap \forall R_3.C_3$

(2)$B \Rightarrow \neg A \sqcup \neg C$

(3)$\neg C \Rightarrow \exists R_2.C_2$

(4)$\neg B \Rightarrow \exists R_1.C_1$

(5)$C \Rightarrow \exists R_1.C_1 \sqcup A$

(6)$A \Rightarrow \exists R_1.C_1 \sqcup \exists R_2.C_2$;

$\mathcal{T}_g$ contains the axioms:

(1)$\top \sqsubseteq \exists R_1.C_1 \sqcup \exists R_2.C_2 \sqcup \forall R_3.C_3$

(2)$\top \sqsubseteq \exists R_1.C_1 \sqcup \exists R_2.C_2 \sqcup B \sqcup C$

In fact, axiom (2) in $\mathcal{T}_g$ is redundant. In the original TBox, we already have an axiom $\{\top \sqsubseteq B \sqcup \exists R_1.C_1\}$ which was absorbed into axiom (4) in $\mathcal{T}_u$, and in axiom (2) in $\mathcal{T}_g$ we also have the segment $\{B \sqcup \exists R_1.C_1\}$. According to the formula $(A \sqcap (A \sqcup C) \equiv A)$[12] provided in **Section 3.2**, axiom (2) in $\mathcal{T}_g$ can be ignored. Moreover, axiom (1) in $\mathcal{T}_g$ can also be absorbed by using *role absorption* (see **Section 4.3**). Thus, the above TBox can be completely absorbed as the following,

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;

$\mathcal{T}_u$ contains the axioms:

$(1) \neg A \Rightarrow (B \sqcup C) \sqcap \forall R_3.C_3$

$(2) B \Rightarrow \neg A \sqcup \neg C$

$(3) \neg C \Rightarrow \exists R_2.C_2$

$(4) \neg B \Rightarrow \exists R_1.C_1$

$(5) C \Rightarrow \exists R_1.C_1 \sqcup A$

$(6) A \Rightarrow \exists R_1.C_1 \sqcup \exists R_2.C_2$;

$(7) \text{Domain}(R_3, \exists R_1.C_1 \sqcup \exists R_2.C_2 \sqcup \forall R_3.C_3)$

$\mathcal{T}_g = \phi$

Unlike the *basic heuristic absorption algorithm*, the *extended heuristic absorption algorithm* does not need additional calculation steps. We can see it as an extension to the *primitive definition to complete definition conversion* and the *basic heuristic absorption algorithm*. To an arbitrary TBox, after we apply the above two algorithms, if there are still some axioms left in $\mathcal{T}_g$, we simply apply **Lemma 4.3** and **Lemma 4.4** to the residual axioms in $\mathcal{T}_g$. If necessary, we apply *role absorption* as well.

It worth to mention that the *extended heuristic absorption algorithm* may not terminate in some cases. The root cause is that during the application of **Lemma 4.3** and **Lemma 4.4**, an additional axiom may be introduced to $\mathcal{T}_g$. If the newly introduced axiom is "simpler"(less disjunctions introduced) than the axiom absorbed,

[12] In this case, we can replace A in the fomula with $\{\top \sqsubseteq B \sqcup \exists R_1.C_1\}$ and C with $\{\exists R_2.C_2 \sqcup C\}$

it would make $\mathcal{T}_g$ smaller. Otherwise, it may make $\mathcal{T}_g$ more complicated and cause non-termination eventually. We leave the topic of heuristically applying **Lemma 4.3** and **Lemma 4.4** to our future work.

Chapter 5

RACER Preprocessor Implementation

In the previous sections we have defined the notion of a *valid absorption*. We also have discussed how to explore absorption to optimize reasoning tasks. In addition to that, we presented several improved absorption algorithms. In this section, we discuss the problem of how to implement these algorithms for real applications. Moreover, we also discuss the problem how to "connect" the newly implemented algorithms with reasoners such as RACER.

To achieve this goal, we will first discuss the system architecture of a RACER preprocessor designed to compare the performance of different absorption algorithms. Then, based on this architecture, we introduce the logic representation and logic operation using the OO model to implement the normalized TBox data structure proposed in **Chapter 3**. The test result and conclusion are discussed in the next chapter.

5.1 Implementation

5.1.1 Architecture

To check the effectiveness of the newly developed algorithms, we compared them with the ones currently employed by RACER. Firstly, we used a customized RACER version with a disabled absorption module. We developed a new external absorption module (as a stand-alone program) by implementing the algorithms described above. Each of our test KBs is processed by the external absorption module. Its output is used as input to the customized RACER version. To test how different absorption algorithms could affect the reasoning performance, we developed a configuration panel. Together with the above mentioned modules, we constructed a preprocessor for RACER.

The reasoning system architecture with preprocessor is shown as follows:

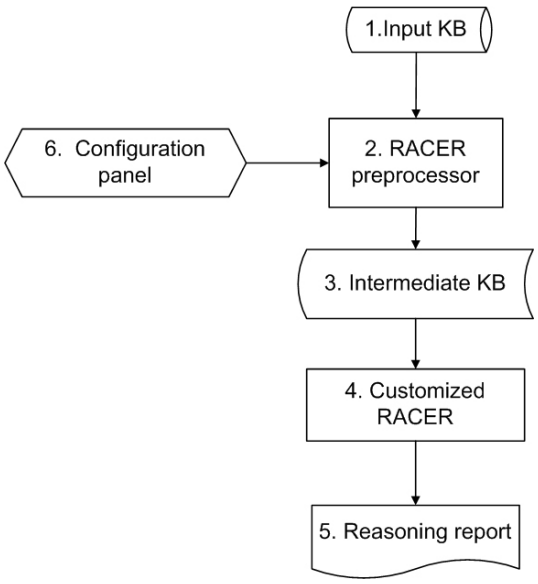

Figure 11: Architecture of the reasoning system with preprocessor

5.1.2 Interface Design with RACER

We decided to use an intermediate file as the interface between the RACER preprocessor and RACER. There are two reasons for this. One reason is that RACER is implemented in LISP. RACER's internal data structures are very different from those of RACER preprocessor which has been developed in JAVA. The other reason is the possible flexibility of integrating it with other systems. Thus, to keep the independence of these two systems, the best solution is to use an intermediate file.

5.1.3 Knowledge Representation by Using OO Design

The OO (object oriented) data model is also suitable to represent knowledge. In fact, the *inheritance* in OO model is very suitable to represent the IS-A relationship in a DL based KB. If it is designed properly, the other attributes in OO model such as encapsulation, polymorphism [12] can also be employed to better represent a KB model.

A typical OO based knowledge representation model applied in our system is shown in Figure 12. In this figure *AtomicConcept, CompoundConcept, NumberRestrictionConcept,* and *RestrictionConcept* are all *concepts* since they all inherit from the **Class** [3] *AbstractConcept* which is an implementation of the **Interface** [3] *Concept*. In addition, we use *attributes* and *methods* of each **Class** to represent their differences and the available operations that could be performed. For example, the results of the methods such as *negation, disjunction* and *conjunction* are all *Concept* even though the implementation of those operations might be different for different *Concepts*. Similarly, the **Interface** *Role* is an abstraction of the *atomic role* in DL which is a required attribute for the concepts *RestrictionConcept* and *NumberRestrictionConcept*. A *Role* also has some attributes such as *domain* and *range*.

From the above description, it is obvious that the above model exactly meets the requirement to represent the $\mathcal{ALCN}$ *Description Language* [see **Chapter 2**].

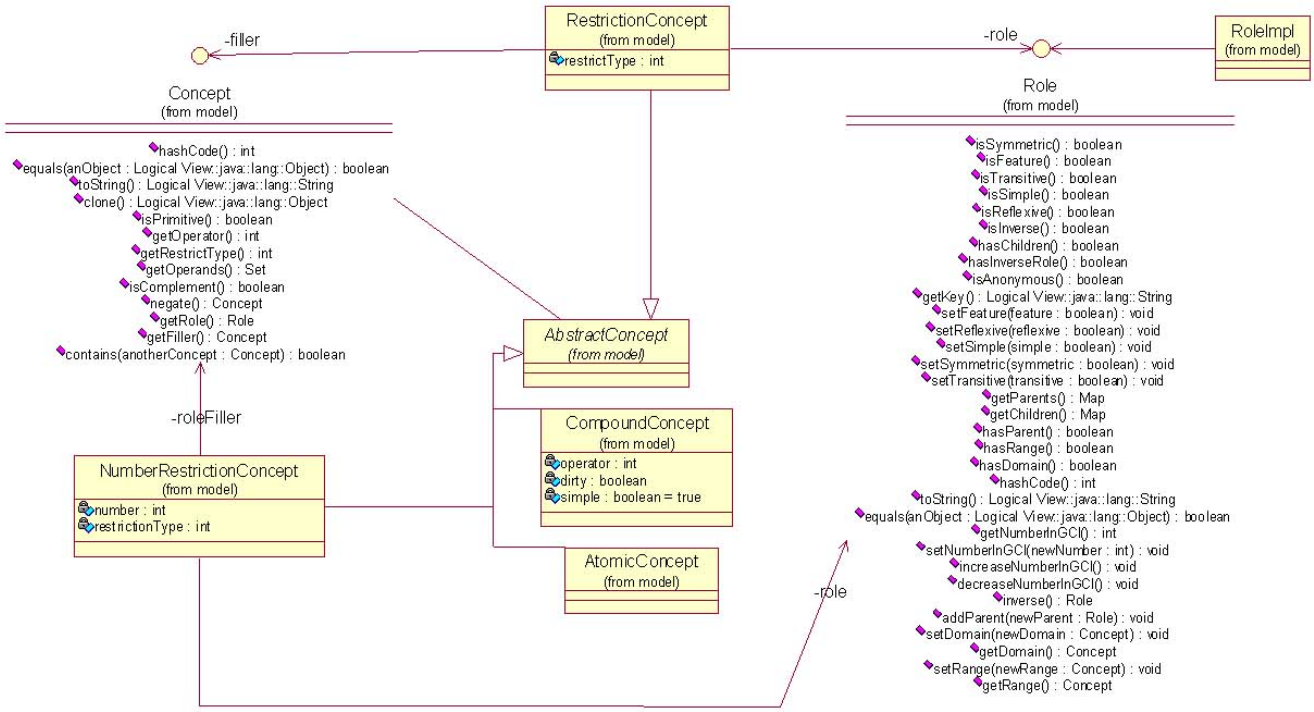

Figure 12: UML of concept/role representation using OO model

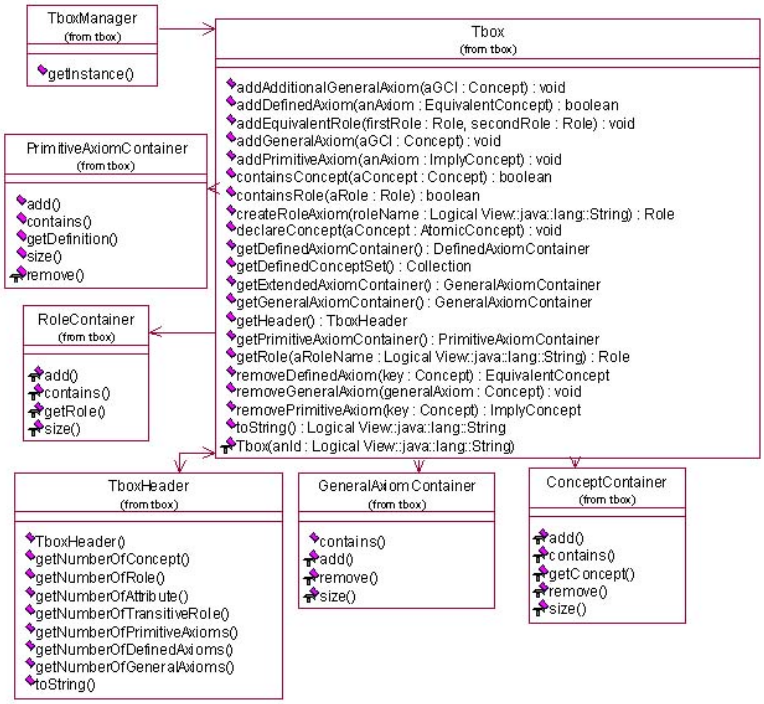

Figure 13: OO model for representing TBoxes

So far, we have discussed the $\mathcal{ALCN}$ *Description Language* representation by using the OO model. To apply the OO model to solve reasoning problems with respect to axioms, we need a model to represent *TBoxes* as well. The Figure 13 shows the OO design model we used in our system.

A TBox can be considered as a container containing a series of containers such as *RoleContainer* which contains *rolenames*, *ConceptContainer* which contains all *conceptnames*, *PrimitiveAxiomContainer* which contains all absorbable axioms, and *GeneralAxiomContainer* which contains all GCIs. Each container is in fact a *Set* [3].

5.1.4 Performance Considerations

Since performance is one of our main considerations, and the OO model emphasizes on the simplification of representation and implementation, the trade off between

implementation and performance has to be considered.

To improve the system performance, we applied two main technique: *hashing* and *caching*. To apply the hashing technique, we designed a hash code generation algorithm for each possible object of interest to accelerate the searching speed in a set container. For caching, we keep an instance for possible reuse in memory to prevent *garbage collection* [3] by the JAVA system.

Chapter 6

Case Study

In this chapter, we first illustrate the algorithms we developed in **Chapter 4** by using some example cases. To check the effectiveness of these newly developed algorithms, we compare them with the ones currently employed by RACER through the preprocessor we introduced in last chapter. Each of our test KBs is processed by the external absorption module. Its output is used as input to the customized RACER version. We also process each original test KB with the standard RACER version. In our graphs we compare for each test KB the TBox classification time of the customized RACER version (denoted as "enhanced") with the standard version of RACER (denoted as "normal").

6.1 Absorb Primitive Definition to Complete Definition

The first pattern we are going to discuss is the kind of TBox $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$ that has the following format (assume A,B,C,D are all atomic concepts):

$$\mathcal{T}_u = \{A \Rightarrow B; C \Rightarrow B; B \Rightarrow A \sqcup C\}$$
$$\mathcal{T}_g = \{\top \sqsubseteq A \sqcup C \sqcup \exists R.D\}$$

The axioms in $\mathcal{T}_g$ can not be absorbed by using the *standard absorption algorithms* illustrated in **Chapter 4** since both A and C already occur on the left-hand side in

$\mathcal{T}_u$. However, by converting $\mathcal{T}_u$ into $\mathcal{T}_u = \{\neg B \Rightarrow (\neg A \sqcap \neg C); B \Rightarrow A \sqcup C\}$[see formula (11) in section 3.2], the above mentioned TBox therefore can be easily absorbed into the following TBox $\mathcal{T}'$:

$$\mathcal{T}_u' = \{B \Rightarrow (A \sqcup C); \neg B \Rightarrow \neg A \sqcap \neg C; \neg A \Rightarrow C \sqcup \exists R.D\}$$
$$\mathcal{T}_g' = \phi$$

The above example can be seen as a kind of pattern which can be completely absorbed by the "enhanced" absorption algorithm. To compare the effectiveness of the "enhanced" absorption algorithm with the "normal" absorption algorithm, one of the best way is to compare the reasoning performance by replicating the same pattern.

The classification time are as follows:

No. of repeated patterns	20	30	50	65	80	95
Normal absorption (s)	2.57	17.80	64.89	174.33	380.31	732.67
Enhanced absorption (s)	0.06	0.10	0.18	0.23	0.30	0.39

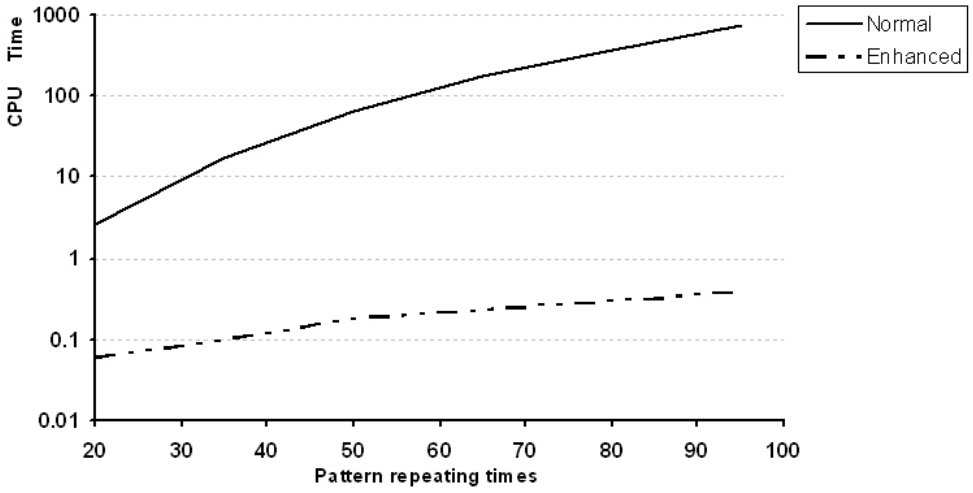

Figure 14: Classification time for *primitive definition to complete definition*

In the above example, we absorbed three primitive axioms from $\mathcal{T}_u$ into two complete concept definition axioms in $\mathcal{T}_u$ in order to absorb more axioms in $\mathcal{T}_g$. It is intuitive to come up with the idea of absorbing more primitive axioms into two concept definition axioms. Let us consider the following example.

A TBox $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$.

$\mathcal{T}_u$ contains the axioms:

(1) $B \Rightarrow \exists R_1.M_1$

(2) $C \Rightarrow \exists R_2.M_2$

(3) $D \Rightarrow \exists R_3.M_3$

And $\mathcal{T}_g$ contains the axioms:

(1) $\top \sqsubseteq \neg A \sqcup \neg B$;

(2) $\top \sqsubseteq A \sqcup B \sqcup C \sqcup D$;

(3) $\top \sqsubseteq \neg A \sqcup \neg C$

(4) $\top \sqsubseteq \neg A \sqcup \neg D$

As we discussed in the previous section, the traditional absorption algorithm can not fully absorb all axioms. By applying the enhanced algorithm, we are able to absorb it into the following format:

$\mathcal{T}_u$ contains:

(1) $B \Rightarrow \exists R_1.M_1$

(2) $C \Rightarrow \exists R_2.M_2$

(3) $D \Rightarrow \exists R_3.M_3$

(4) $A \Rightarrow \neg B \sqcap \neg C \sqcap \neg D$

(5) $\neg A \Rightarrow B \sqcup C \sqcup D$

$\mathcal{T}_g = \phi$

Now let us check the result:

No. of repeated patterns	20	30	50
Normal absorption (s)	16.52	133.89	521.96
Enhanced absorption (s)	0.07	0.12	0.20

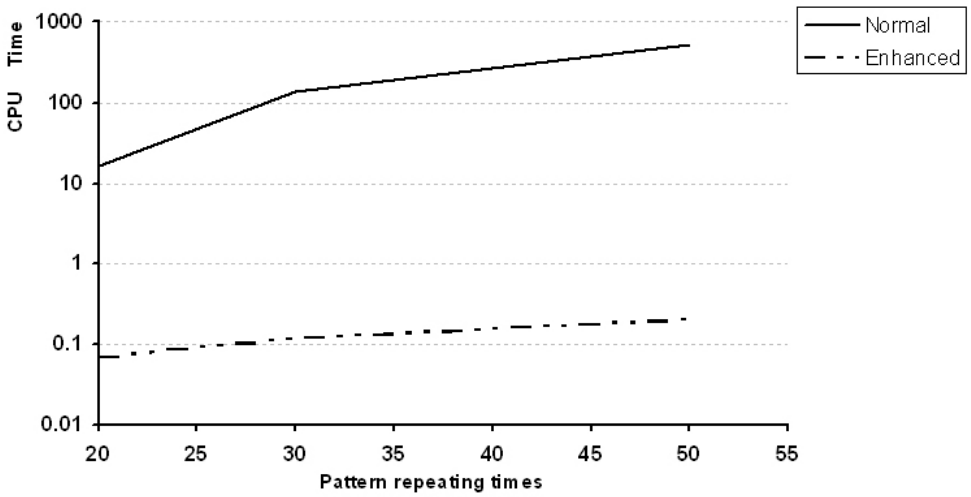

Figure 15: Classification time for *primitive definition to complete definition* – to absorb more primitive axioms

Compared with the result in Figure 14, a better performance improvement is achieved.

6.2 Heuristic Absorption

6.2.1 Basic Heuristic Absorption Algorithm

As we have discussed in **Chapter 4**, this absorption algorithm is based on the statistics about the occurrence of atomic concepts. The goal is to absorb as many axioms as possible from $\mathcal{T}_g$. The heuristic criterion depends on the positive and negative counters of an atomic concept. If one of those counters is zero, then this atomic concept will have a higher absorption priority than others. In addition, if two atomic concepts have one counter with a 0 value, then the concept with the higher value than the other will have a higher priority for absorption; if all counters are the same for different concepts, then the absorption priority for these two atomic concepts will be equal.

Suppose we have the following TBox $\mathcal{T}$ which consists of the following GCIs:

$$\top \sqsubseteq \neg A \sqcup B$$

$$\top \sqsubseteq \neg C \sqcup \neg D \sqcup \exists R_1.C_1 \sqcup R_2.C_2$$

$$\top \sqsubseteq \neg D \sqcup K$$

$$\top \sqsubseteq \neg M \sqcup N$$

$$\top \sqsubseteq C \sqcup D \sqcup \exists R_1.C_1 \sqcup M \sqcup A$$

First let us absorb the above TBox by applying the *standard absorption algorithm*. For the detailed steps regarding to *standard absorption algorithm*, please refer to Section 4.5.1. The result is shown as follows:

$\mathcal{T}_u$:

$$A \Rightarrow B$$

$$C \Rightarrow \neg D \sqcup \exists R_1.C_1 \sqcup R_2.C_2$$

$$D \Rightarrow K$$

$$M \Rightarrow N$$

$\mathcal{T}_g$:

$$\top \sqsubseteq C \sqcup D \sqcup \exists R_1.C_1 \sqcup M \sqcup A$$

For the last axiom, since C, D, M, A already occur on the left-hand side of $\mathcal{T}_u$, this axiom has to remain in $\mathcal{T}_g$.

By following the heuristic absorption algorithm, however, we may completely absorb this KB with the following procedure for this TBox:

Step 1: Compute the statistics of T_g for each atomic concept. The result of the above example (format is $A(c_p, c_n)$) is as follows:

A(0,1); B(1,0);C(1,1);D(1,2);M(1,1); N(1,0); K(1,0);$C_2(0,0)$; $C_1(0,0)$

Divide the above statistics into two groups (we disregard the concepts where both c_p and c_n are 0):

B(1,0); N(1,0); K(1,0)

A(1,1); C(1,1);D(1,2);M(1,1)

Step 2: We give the concepts in the first group a higher priority. By applying this heuristic criterion, the concepts B, N, K have the same absorption priority. Suppose we absorb into B first, then we have the following absorption result:

T_u contains the following axioms:

$\neg B \Rightarrow \neg A$

T_g contains the following axioms:

$\top \sqsubseteq \neg C \sqcup \neg D \sqcup \exists R_1.C_1 \sqcup R_2.C_2$

$\top \sqsubseteq \neg D \sqcup K$

$\top \sqsubseteq \neg M \sqcup N$

$\top \sqsubseteq C \sqcup D \sqcup \exists R_1.C_1 \sqcup M \sqcup A$

We repeat step 1 and step 2. The statistics of T_g now becomes:

A(1,0); N(1,0); K(1,0)

C(1,1);D(1,2);M(1,1)

Among A, N, K, suppose we absorb into A this time since these three atomic concepts all have the same priority upon our criterion. The TBox now becomes:

T_u contains the following axioms:

$\neg B \Rightarrow \neg A$

$\neg A \Rightarrow C \sqcup D \sqcup \exists R_1.C_1 \sqcup M$

T_g contains the following axioms:

$\top \sqsubseteq \neg C \sqcup \neg D \sqcup \exists R_1.C_1 \sqcup R_2.C_2$

$\top \sqsubseteq \neg D \sqcup K$

$\top \sqsubseteq \neg M \sqcup N$

The consequent statistics of T_g is updated to:

N(1,0); K(1,0); M(0,1); C(0,1);D(0,2);

We repeat step 1 and step 2 to absorb D and M. The ultimate absorption result is the following:

T_u contains the following axioms:

$\neg B \Rightarrow \neg A$

$\neg A \Rightarrow C \sqcup D \sqcup \exists R_1.C_1 \sqcup M$

$D \Rightarrow K \sqcap (\neg C \sqcup \exists R_1.C_1 \sqcup R_2.C_2)$

$M \Rightarrow N$

$T_g = \phi$

The test result of the above absorption is as the following:

Axiom pairs	20	35	50	65	80
Classic absorption (s)	4.89	35.14	131.13	350.37	773.31
Heuristic absorption (s)	0.14	0.3	0.54	0.76	1.02

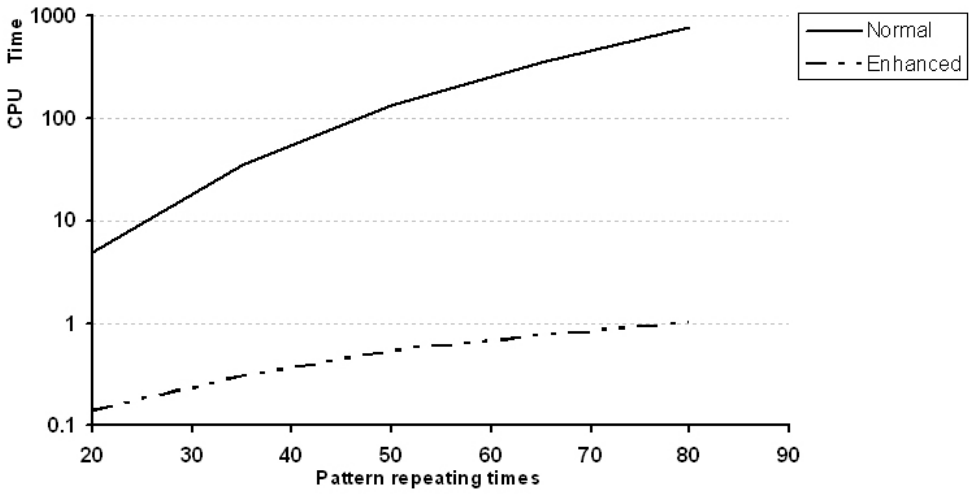

Figure 16: Classification time for the *basic heuristic absorption algorithm*

6.2.2 Extended Heuristic Absorption Algorithm

Based on the discussion in **Section 4.5.4**, the *extended heuristic absorption algorithm* can be applied if there are still some GCIs left in $\mathcal{T}_g$ after the application of *primitive concept definition to complete concept definition conversion* and *basic heuristic*

absorption algorithm. To evaluate the effectiveness of such kind of absorption, we conducted some test based on the example KB discussed in the **Section 4.5.4**.

The original TBox is as the following where $A, B, C \in NC$:

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;

$\mathcal{T}_u = \phi$;

$\mathcal{T}_g$ contains the axioms:

$\top \sqsubseteq \neg A \sqcup \neg B \sqcup \neg C$,

$\top \sqsubseteq A \sqcup B \sqcup C$,

$\top \sqsubseteq B \sqcup \exists R_1.C_1$,

$\top \sqsubseteq C \sqcup \exists R_2.C_2$,

$\top \sqsubseteq A \sqcup \forall R_3.C_3$

To simplify our discussion, please refer to Section 4.5.4 for the detailed absorption steps. After absorption, we have the following result:

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;

$\mathcal{T}_u$ contains the axioms:

(1)$\neg A \Rightarrow (B \sqcup C) \sqcap \forall R_3.C_3$

(2)$B \Rightarrow \neg A \sqcup \neg C$

(3)$\neg C \Rightarrow \exists R_2.C_2$

(4)$\neg B \Rightarrow \exists R_1.C_1$

(5)$C \Rightarrow \exists R_1.C_1 \sqcup A$

(6)$A \Rightarrow \exists R_1.C_1 \sqcup \exists R_2.C_2$;

(7)Domain$(R_3, \exists R_1.C_1 \sqcup \exists R_2.C_2 \sqcup \forall R_3.C_3)$

$\mathcal{T}_g = \phi$

The experimental result for the above absorption is as follows:

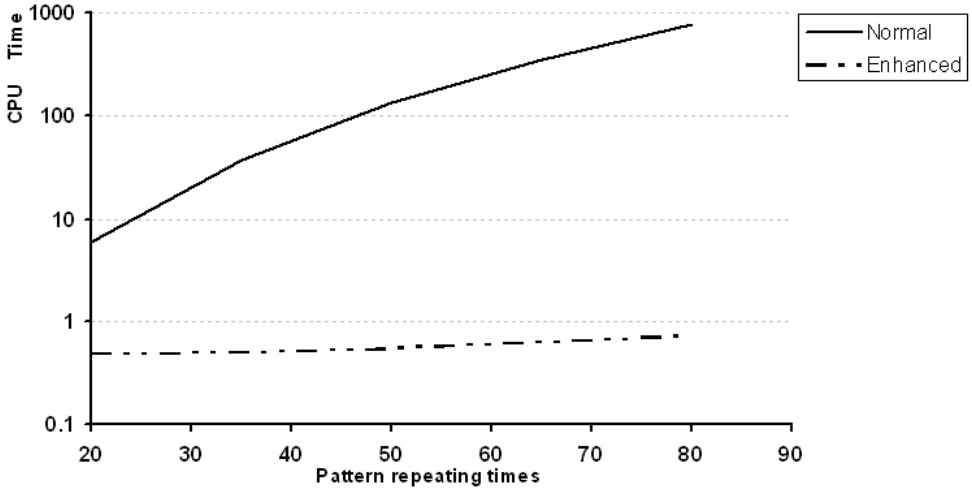

Figure 17: Classification time for the *extended heuristic absorption algorithm*

Pattern repeating times	20	35	50	65	80
Normal absorption (s)	5.87	37.66	133.29	357.75	766.10
Enhanced absorption (s)	0.48	0.50	0.56	0.64	0.73

Chapter 7

Future Work

7.1 The Criteria for Best Absorption

In previous discussion, we have proposed the criterion for the "best absorption" based on experimental result. The underlying assumption for this criterion is that the reasoning performance is mostly affected by the disjunctions in $\mathcal{T}_g$ during *lazy unfolding*. However, in practice, even though a TBox can be completely absorbed, different absorption results may cause different performances. Consider a very simple example. The TBox $\mathcal{T}$ can be easily rewritten in three formats:

Format 1: All axioms remain unabsorbed, i.e. all axioms are in $\mathcal{T}_g$, then the TBox contains the following axioms:

(i)$(\top \sqsubseteq \neg B \sqcup A)$

(ii)$(\top \sqsubseteq \neg C \sqcup A)$

(iii)$(\top \sqsubseteq \neg D \sqcup A)$

(iii)$(\top \sqsubseteq \neg E \sqcup D)$

(iv) $(\top \sqsubseteq \neg F \sqcup C)$

(iv)$(\top \sqsubseteq \neg G \sqcup B)$

Format 2: All axioms are absorbed, the absorption sequence is exactly the same as the concept hierarchy:

(i)$(B \Rightarrow A)$

(ii)$(C \Rightarrow A)$

(iii)$(D \Rightarrow A)$

(iii)$(E \Rightarrow D)$

(iv)$(F \Rightarrow C)$

(iv)$(G \Rightarrow B)$

Format 3:All axioms are absorbed, the absorption sequence is exactly the inverse as the concept hierarchy:

(i)$(\neg A \Rightarrow \neg B \sqcap \neg C \sqcap \neg D)$

(iii)$(\neg D \Rightarrow \neg E)$

(iv)$(\neg C \Rightarrow \neg F)$

(iv)$(\neg B \Rightarrow \neg G)$

By duplicating the same pattern 300 times, the classification performance of each format is shown as the following:

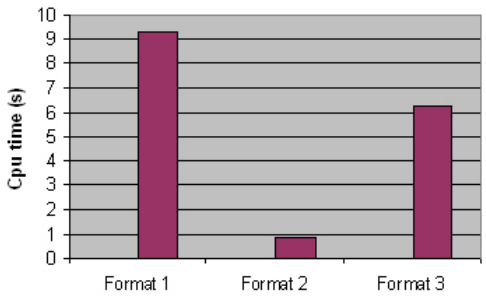

Figure 18: Classification time for different absorption format

From the above test result, we can see that even though in format 2 and format 3 all axioms are completely absorbed, the reasoning performances are quite different. Therefore, finding out the criteria for the "best" absorption is still one of our future research directions.

7.2 Basic Heuristic Absorption

As we have seen in the examples during the discussion about heuristic absorption, choices still exist upon the heuristic criterion we setup. In addition, the underlying idea of our heuristic absorption algorithm is to absorb as many GCIs as possible which is slightly different from the criterion we defined for the "best" absorption.

Both a better criterion for heuristic priority setting and an algorithm based on the number of disjunctions in $\mathcal{T}_g$ are our future research directions.

7.3 Extended Heuristic Absorption

We have discussed the *extended heuristic absorption algorithm* which is able to absorb GCIs from $\mathcal{T}_g$ to $\mathcal{T}_u$ regardless of the left-hand side atomic concept sign restriction[1] (see 4.5.4).

Theoretically, we claim that all KBs can be completely absorbed except the kind of axioms in the form of $\{\exists R_1.C_1 \sqcup \exists R_2.C_2\}$ by applying the *extended heuristic absorption algorithm*. In this way, *lazy unfolding* is applicable to most of axioms in a KB. Therefore, a significant performance improvement should be achieved in application.

Unfortunately, directly applying the *extended absorption algorithm* to an arbitrary KB may cause nontermination. The KB $BCS4$ and $BCS5$ are typical examples for such kind of KB. The root cause of it is due to the newly introduced axiom in $\mathcal{T}_g$ during absorption. As we know, by applying **Lemma 4.3** and **Lemma 4.4** to absorb axioms into both negative and positive concept names in $\mathcal{T}_u$, a trade off is that a new disjunctive axiom has to be added into $\mathcal{T}_g$. If after absorption, $\mathcal{T}_g$ contains less axioms or less disjunctions, we say the size of $\mathcal{T}_g$ is *reduced*. In fact, not all absorptions reduce the size of $\mathcal{T}_g$. Thus, nontermination may occur.

To solve this problem, more heuristic algorithms need to be developed in future

[1] The restriction of not allowing both negative and positive concept names occur on the left-hand side of $\mathcal{T}_u$ at the same time.

work to deal with the newly introduced axiom to $\mathcal{T}_g$.

7.4 Absorption of more Expressive DL

Up to now, all our discussions regarding to absorption are conservatively restricted to the $\mathcal{ALCN}$ DL, but we claim that the results we achieved in the previous sections are also applicable to very expressive DLs and OWL-DL. To apply the absorption algorithms to very expressive DLs and OWL-DL is one of our future research directions.

Chapter 8

Contribution

We proposed criteria for the "best" absorption based on experimental experience. Then, we introduced novel heuristic absorption algorithms. We have demonstrated how these algorithms work, and how they affect the reasoning performance. In addition, we have shown that some restrictions applied in the absorption algorithms of RACER could be eliminated. Therefore, the absorption algorithms are able to be applied to more general axioms.

We have also implemented the heuristic algorithms by incorporating the optimizations known from the RACER reasoner. We have illustrated their effectiveness by analyzing the reasoning performance of RACER when classifying benchmark KBs. The analysis shows that, not only are the new techniques highly effective, but also the reasoning performance is not significantly affected by the order and format of the axioms occurring in a KB.

Bibliography

[1] Franz Baader and Werner Nutt. *THE DESCRIPTION LOGIC HANDBOOK:Theory, implementation, and applications*, chapter 2: Basic Description Logics, pages 47–100. Cambridge University Press, 2003.

[2] Diego Calvanese and Giuseppe De Giacomo. *THE DESCRIPTION LOGIC HANDBOOK:Theory, implementation, and applications*, chapter 5: Expressive Description Logics, pages 184–225. Cambridge University Press, 2003.

[3] Gary Cornell Cay S Horstmann and Cay S Forstmann. *Core Java 2*. Prentice Hall PTR, 1st edition, 2002. ISBN 0130471771.

[4] Bernhard Hollunder Bernhard Nebel Franz Baader, Enrico Franconi and Hans-Jurgen Profitlich. An empirical analysis of optimization techniques for terminological representation systems, or: Making kris get a move on. int. conf. on the principles of knowledge representation and reasoning. *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR92)*, pages 270–281, 1992.

[5] Volker Haarslev and Ralf Möller. High performance reasoning with very large knowledge bases. *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence*, pages 161–166, 2001.

[6] Volker Haarslev and Ralf Möller. Racer user's guide and reference manual version 1.6. Technical report, University of Hamburg, Computer Science Department, Germany, 2001.

[7] I. Horrocks and S. Tobies. Optimisation of terminological reasoning. *Proc. of the 2000 Description Logic Workshop*, pages 183–192, 2000.

[8] I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference*, pages 285–296, 2000.

[9] Ian Horrocks. *THE DESCRIPTION LOGIC HANDBOOK:Theory, implementation, and applications*, chapter 9: Implementation and Optimisation Techniques, pages 313–355. Cambridge University Press, 2003.

[10] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *Proc. of the 2003 Description Logic Workshop (DL 2003)*, 81:1–8, 2003.

[11] Daniele Nardi and Ronald J. Brachman. *THE DESCRIPTION LOGIC HANDBOOK:Theory, implementation, and applications*, chapter 1: An Introduction to Description Logics, pages 5–44. Cambridge University Press, 2003.

[12] Meilir Page-Jones and Larry L Constantine, editors. *Fundamentals of Object-Oriented Design in UML*. Addison-Wesley Professional, 1999. ISBN 020169946X.

[13] Steve Reeves and Michael Clarke. *Logic For computer science*. Addison-Wesley, 1990.

[14] Dmitry Tsarkov and Ian Horrocks. Efficient reasoning with range and domain constraints. *Proc. of the 2004 Description Logic Workshop(DL 2004)*, pages 41–50, 2004.